
(2)

RL-TR-91-406
Final Technical Report
December 1991

# AUTOMATIC ACCESS EXPERIMENT



Software Productivity Solutions

*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*





Rome Laboratory
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-91-406 has been reviewed and is approved for publication.

APPROVED: Patrick K McCabe

PATRICK K. MCCABE
Project Engineer

FOR THE COMMANDER: Thadeus J Domurat

THADEUS J. DOMURAT
Technical Director
Intelligence & Reconnaissance Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL(IRDD) Griffiss AFB, NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188



| 1. AGENCY USE ONLY (Leave Blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | December 1991 | Final Feb 89 - Nov 91 |

**4. TITLE AND SUBTITLE**
AUTOMATIC ACCESS EXPERIMENT

**5. FUNDING NUMBERS**
C - F30602-89-C-0003
PE - 62702F
PR - 4594
TA - 16
WU - 33

**6. AUTHOR(S)**
---

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Software Productivity Solutions
Indialantic FL 32903

**8. PERFORMING ORGANIZATION REPORT NUMBER**
N/A

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
Rome Laboratory (IRDD)
Griffiss AFB NY 13441-5700

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
RL-TR-91-406

**11. SUPPLEMENTARY NOTES**
Rome Laboratory Project Engineer: Patrick K. McCabe/IRDD/(315) 330-2171

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** (Maximum 200 words)

AAE was a research and development program which attacked one of the most important problems experienced by large organizations in the 1990's; the problem of maintaining and using the many diverse data storage and retrieval systems on which the organization has come to rely on. There are three facets to this problem. One, data is stored in many different computer systems. Two, it is stored in many different management systems. Three, data is stored in many different formats. While retrieving data, a user must consider these varied ways data is stored and have knowledge of the information's location and the commands needed to retrieve the separately stored data.

The AAE project conducted research to find methods which would provide transparent access to distributed, heterogeneous, multimedia data. Using principles uncovered by the research, a prototype program known as the Automated Transparent Object Management System (ATOMS) was developed which applies and demonstrates these principles.

ATOMS is a prototype system, but one which can be scaled up to production performance requirements without deviating from the existing design.

**14. SUBJECT TERMS** Multimedia, Heterogeneous Database Access, Correlation and Fusion, Entity Relationship Model, Network Administration

**15. NUMBER OF PAGES** 88

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UL |

## Document Figures

## Document Tables

# Document Introduction

This document is the Final Technical Report for the Automated Access Experiment (AAE) program, under contract number F30602-89-C-0003 for Rome Laboratory.

AAE was a research and development program which attacked one of the most important problems experienced by large organizations in the 1990s; the problem of maintaining and using the many diverse data storage and retrieval systems on which the organization has come to rely. There are three facets to this problem. One, data is stored in many different computer systems. Two, it is stored in many different management systems. Three, data is stored in many different formats. While retrieving data, a user must consider these varied ways data is stored and have knowledge of the information's location and the commands needed to retrieve the separately stored data.

The AAE project conducted research to find methods which would provide transparent access to distributed, heterogeneous, multimedia data. Using principles uncovered by the research, a prototype program known as the Automated Transparent Object Management System (ATOMS) was developed which applies and demonstrates these principles .

ATOMS is a prototype system, but one which can be scaled up to production performance requirements without deviating from the existing design. Eight specific goals which were formulated and used in the development of ATOMS:

1) Provide transparent access to distributed, heterogeneous, multimedia data.
2) Provide powerful user defined data views and queries.
3) Support data correlation and fusion.
4) Control user access to data and identify classified data
5) Provide a modern bit mapped user interface
6) Build a system to demonstrate the concept of modeling heterogeneous, multimedia data using the Entity/Relationship Model.
7) Design a system which can be scaled up and enhanced without extensive reconstruction of the existing software.
8) Design the system for portability

The report discusses the development process of the system and explains how ATOMS key design characteristics meet the eight specific goals defined during the research phase. This Final Technical Report is organized into the following sections:

1. Document Introduction
2. ATOMS Architectural Design
3. ATOMS Data Servers
4. ATOMS Development Process
5. Future System Development Recommendations
6. Running ATOMS and its Network Servers
7. AAE Program Studies and Metrics

# ATOMS Architectural Design

As discussed in the Introduction, intelligence organizations today experience difficulty in fully utilizing the large volume of data they collect. This problem is true partly because data is stored in many different computer systems, in many different storage management systems, and in many different formats. It is difficult and inconvenient to manually retrieve separately stored information about objects of interest, requiring knowledge of both the information's location, and the commands needed to retrieve it.

The major goal of ATOMS is to provide transparent access to distributed, heterogeneous, multimedia data. ATOMS allows knowledgeable database administrators to set up schemas which describe the location and content of databases residing on a TCP/IP network. Using the schema information, ATOMS enables users to transparently access any data described by the database administrators, regardless of where it is stored or what database manager is needed to access it. ATOMS provides discretionary access control to data. Access to data can be granted or revoked on an individual user basis. ATOMS presents a single, consistent user interface to all the data described in its schema.

ATOMS is not a data repository in itself. It is an access mechanism for existing data repositories. A source of existing information is known within ATOMS as a *data source*. Each data source is encapsulated by a TCP/IP network server process known as a *data server*. ATOMS communicates with data servers using the ATOMS Entity-Relationship (ER) language. This language defines data, entity, and relationship types, schemas, views, and queries. There are currently several different types of data servers within ATOMS, and because there is a standard interface, more can be constructed and added without impacting the rest of the system. ATOMS External Architecture is shown in Figure 1.

The ATOMS user role known as the *local database administrator* is responsible for modeling the data encapsulated by the data servers using the *local schema* construct. Each local schema defines an ER model of the data stored by a single data server. Multiple local schemas may be defined for a single data server, however, a single local schema may reference only one data server. The local schema specifies a mapping between the data source and the ATOMS ER language.

A data server is responsible for translating between the native data manipulation language of the data source and the ATOMS ER language. All data servers translate ATOMS ER queries to the native query language, and then translate data in the native format to the ATOMS ER language types. Together the data servers and their local schemas solve the problems of handling multimedia data and interfacing to heterogeneous databases. The local schemas present an interface where all data is modeled using the ATOMS ER language regardless of how it is physically stored by the data source.

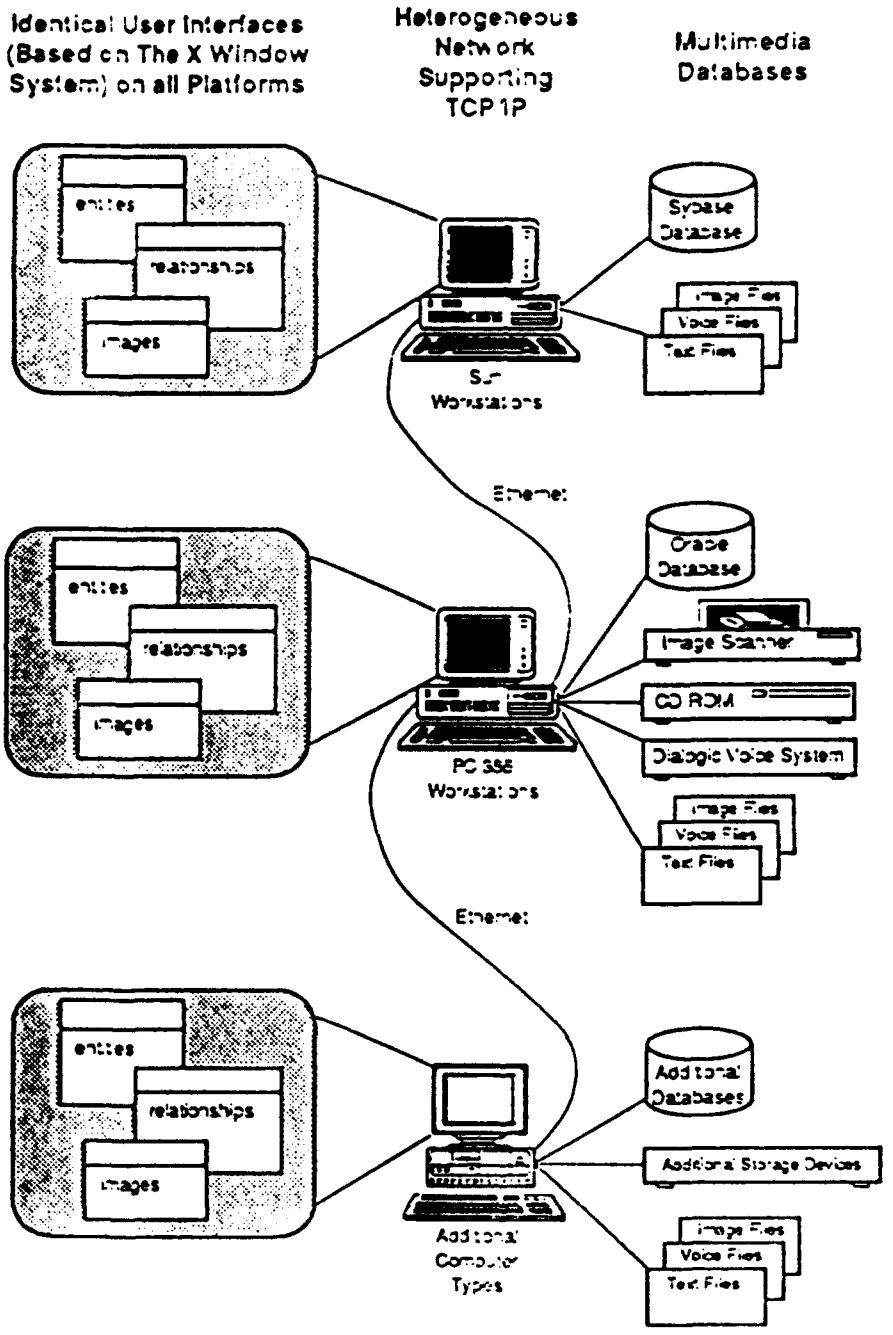


Figure 1. ATOMS External Architecture

The extensibility of ATOMS to other database managers requires that no special semantics and syntactic structures be implemented to support a particular database above the data server level. If database specific types are visible at the local schema level, other than as un-interpreted strings, then the local schema level of ATOMS will have to be changed to implement another database manager.

Each local schema describes one database of information stored by a single data server. The ATOMS user role known as the *network database administrator* is responsible for mapping all the separate locals schemas into the *network schema*. The network schema models the information in the local schemas in a single unified schema in which all entity and relationship types have unique names within their subschemas. The network schema is used to present a single consistent interface to all the data available on the ATOMS network. It solves the problem of accessing distributed data. The network schema is divided in subschemas to make it easier to administer.

Separating the problems of accessing heterogeneous, multimedia databases (in the local schemas) and accessing distributed data (in the network schema) simplifies the design and use of the ATOMS system.

All data access in ATOMS is done in the context of *views*. A view is a collection of entity and relationship types which may be pulled from one or more different network subschemas. An ATOMS user can create as many views as they wish, where each view can be used to provide access to different data areas of interest.

A view may contain entity and relationship types which map directly to entity and relationship types defined in different network subschemas. This allows visibility to the data modeled by the network database administrator. A view may also contain entity types which map to multiple network subschema entity types. A view entity type consists of one or more consecutive sources, each of which may have zero or more alternate sources. A consecutive source is a mapping to a network entity type which may include one or more of its attributes. A view entity type with multiple consecutive sources combines into one entity type instances of the same conceptual entity type that are stored in different data sources (i.e., the view forms a union of the consecutive sources). This allows all the instances of a particular conceptual entity type to be retrieved, browsed, and searched without regard to where they are stored or how they are retrieved. By defining a view entity type with consecutive sources, all the data on a network about a particular conceptual entity (for instance ships) can be searched using one operation.

A view entity type with multiple consecutive sources is assembled by retrieving each consecutive source in order and combining the resulting resulting instances into a continuous stream. This capability is conceptually different than connecting distinct entity types using a relationship type.

Each consecutive source of a view entity type may have zero or more alternate sources associated with it. An alternate source is a mapping to a network entity type which provides the same information as the consecutive source. Alternate sources allow ATOMS to make use of redundant or similar data when the consecutive source is unavailable. The alternate sources are consulted in a user defined order.

In addition to mapping to network relationship types, views allow the user to define new relationship types between any entity types in a view. This allows the user to define and retrieve their own relationships, in addition to those provided by the network database administrator. Because the domain and range entity types in the new relationships may have multiple consecutive and alternate sources, the new relationship may benefit from these capabilities.

Views provide transparent access to data to the ATOMS user. A user does not need to know where data is physically located, what database system it is stored in, or how to access the data in order to retrieve it in ATOMS. Figure 2 displays ATOMS schema levels and corresponding user roles.

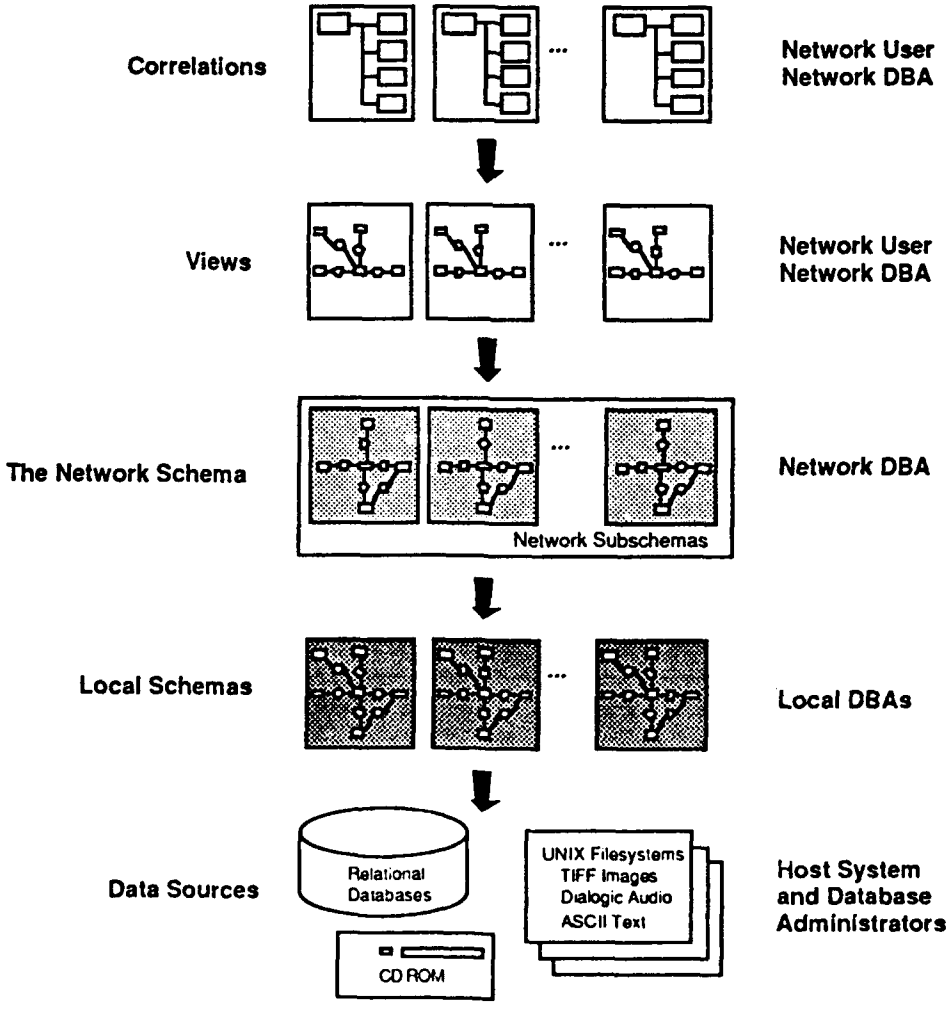


Figure 2. ATOMS Schema Levels and Corresponding User Roles

## Entity/Relationship Modeling

The ATOMS solution to the distributed access problem uses three levels of Entity/Relationship (ER) schema information. The ER model was first defined by Peter Chen [CHE76]. The ER model is a more natural way to express the characteristics and connections of domain data than other data modeling languages such as the Schema Query Language (SQL). Entities and relationships are very natural human concepts, making modeling of complex data relatively straightforward.

The ER model has been seen as a unifying model for data description, because well defined rules exist for transforming an ER model to a hierarchical, network or SQL model; therefore, the ER model is a natural schema representation for a system which integrates heterogeneous database types.

## Schema Element Names

The following schema element names in the ATOMS ER language may be from 1 to 32 characters in length: local schema names, local entity type names, local relationship type names, local attribute names, network subschema names, network entity type names, network relationship type names, network attribute names, view names, query names, correlation set names, correlation names, and user names. Letter case is not significant in comparing names of the listed types. All other string information in ATOMS is stored and used in the letter case in which it was entered, primarily to aid database administrators in accommodating data servers to which letter case is significant.

## Local Schemas

The purpose of a local schema is to define an ER model of a single data source and to provide the information needed to map from the data source's database to the ATOMS E/R Language. A local schema defines the entity and relationship types that model a single data source accessed by a single data server. The local schema definition is not tied to a particular database model. Local schemas may be written for interpretation by relational data servers, filesystem data servers, etc. ATOMS is intended to be extensible to hierarchical, network, CODASYL and other database types. Several mappings between the ER model and other database types have been published. Figure 3 shows an example local schema utilization within ATOMS.

The type of data server accessing the data source modeled by a local schema is not important to ATOMS. All data servers communicate with ATOMS using the same network interface protocol, ATOMS ER language, and query model.

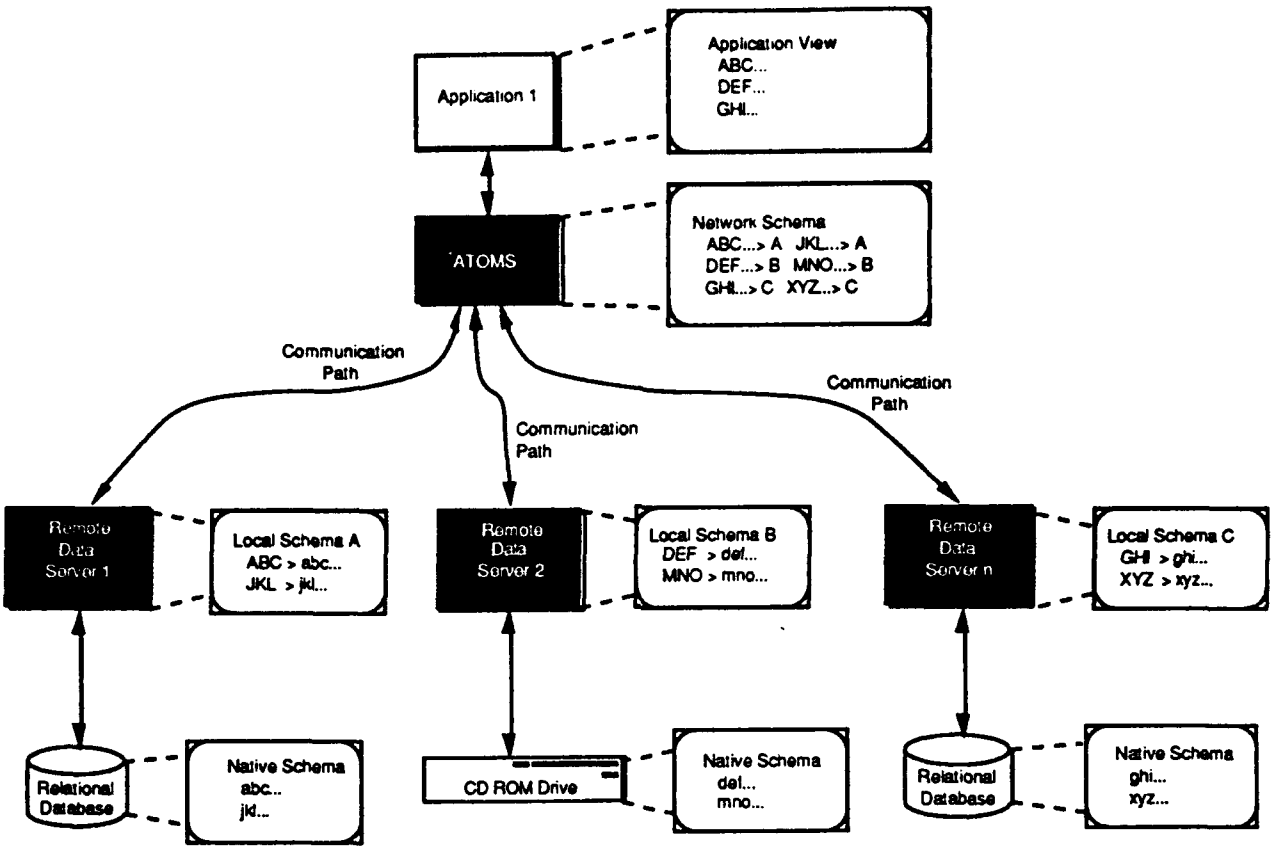


Figure 3. Example Local Schema Utilization

Some data servers may not implement the entire ER language or query model. ATOMS does not check if a local schema uses features not implemented by the data server to which it is tied.

The information content of a local schema is the following:

```
local schema <identifier> is
        access list [ <identifier> ]* ;
        level <string> compartment <string> [ training ]

        database node is <string>
        database server is <string>
        database name is <string>
        database type is <string>
        database user is <string>
        database password is <string>

        [ <local entity type> | <local relationship type> ]*
end local schema
```

### Local Entity Types

A local entity type definition defines a mapping from a named component of a data source in an external database to an ER entity type. The information content of a local entity type definition is the following:

```
local entity <identifier> is
        level <string> compartment <string> [ training ]
        source is <string>
        [ <attribute> : [ key ] <type> from <string> : <string> ]*
end local entity
```

Every local entity type must designate one and only one attribute as the key attribute. For the convenience of the network user the key attribute values should identify the instances of the modeled entity as uniquely and intuitively as possible, although ATOMS does not requires that the key attribute values be unique. The key attribute is used in ATOMS for identifying entity instances which belong to correlations. The key attribute designation is intended to help users who are creating network subschemas in determining which local entity attribute uniquely identifies the entity instances. The key designation can be overridden in the network subschema level, so it functions mostly as a comment to network subschema creators.

The host database system types are recorded because some data servers must know the native database types and they can not always be inferred from an attribute's corresponding ATOMS types. Native database types are treated by ATOMS as un-interpreted strings which are stored and passed to the data servers.

### Local Relationship Types

A local relationship type definition defines a named connection between a *domain* local entity type and a *range* local entity type. Although they are bidirectional, a relationship is defined semantically as originating at the domain entity and connecting to the range entity. Instances of the local relationship type are determined by matching the domain key field of the domain entity type with the range key field of the range entity type. Several criteria for determining if the two field values match are supported. The information content of a local relationship type definition is the following:

```
local relationship <domain> <relation> <range> is
        level <string> compartment <string> [ training ]
        domain key <attribute>  <matching criteria>
        range key <attribute>  <matching criteria>
        [ <attribute> from [ domain | range | relation ]+
<attribute> ]*
end local relationship
```

Local relationship type names consist of the name of the domain entity type, followed by the relation name, followed by the name of the range entity type. Local relationship type name components are separated by a single space. Zero or one domain attributes may be designated as the domain key. Zero or one range attributes may be designated as the range key.

Separating the relationship type name components with spaces may not be compatible with the IRDS standard but is in compliance with typical industry practice, and it avoids the ambiguity present in the needlessly obscure IRDS relationship names.

The domain and range local entity types may be the same (defines a recursive relationship). A recursive relationship may use the same attribute for the domain and range keys. This defines a relationship where each entity instance is related to itself at least. ATOMS does not support the concept of relationship cardinality. Specifying a cardinality is not useful when the relationship data already exists. There is no guarantee that the cardinality specified by the local database administrator is correct, specifying one could be misleading.

A relationship instance exists between domain and range instances if their key values are equal after being transformed by the matching criteria. The ATOMS ER language supports mapping attributes to the relationship (as opposed to the domain or range), however this capability will not be implemented initially because none of the initial data servers support it.

### Matching Criteria

Matching criteria are used by ATOMS in comparing key values when determining relationship membership. The purpose of the various options is to get matches in databases which were created independently and are therefore not entirely consistent. The matching criteria can be used to overcome some differences in format and usage. The information content of the matching criteria is the same in all ATOMS schema levels:

```
match on
      [ exact case | ignore case ]+
      [ range ( <integer> .. <integer> ) ]
      [ translate (  [ '<character>' > [ e | '<character>' ]+ ]#)]
      [ trim [ leading | trailing ] ]
```

For example, to match dates in the form "XX-XX-XX" with dates in the form "XX/XX/XX" use translate ('/' > '-'). To get rid of periods in abbreviations use translate ('.' > e). This will match "U.S.A." with "USA". To remove underscores and dashes use translate ('_' > ' ' | '-' > ' '). The 'e' in this case means the empty

string, which is a string of zero characters, and is different than a string with one or more blanks.

## The Network Schema

The network schema combines the local schemas into a single unified ER schema modeling all the data on the network available to ATOMS. It hides the physical format and location of the data modeled by the local schemas. The network schema allows name collisions between elements in different local schemas to be resolved, and allows the adoption of a consistent naming policy, increasing the illusion of a single centralized database, and thereby improving user comprehension of the available data. The network schema provides a consistent foundation for building the network and user views.

The information content of the network schema is the following:

```
network schema is
      [ <network subschema> ]*
end network schema

network subschema <identifier> is
      access list [ <identifier> ]* ;
      level <string> compartment <string> [ training ]
      [ <network entity type>  |  <network relationship type>  ]*
end network subschema
```

The network schema is composed of subschemas to assist the Network DBA in managing the expected large volume of schema data. When a network subschema is created, its access list contains only the name of the user who created it.

### Network Entity Types

Network entity types map to local entity types, making them visible in the network schema. A network entity type may rename the local entity type to which it maps, and each network attribute may rename the local attribute to which it maps. A network entity type may model one or more of the attributes of a local entity type.

The information content of a network entity type is the following:

```
network entity <identifier> is
      level <string> compartment <string> [ training ]
      source <local schema>.<entity name>
      [ <attribute> [ key ] from <attribute> ]*
end network entity
```

The type of a network attribute is identical to the type of the local attribute to which it maps. A network entity type may have zero or one attributes designated as the key attribute. The key attribute designation is intended to help users who are creating network views in determining which network entity attribute uniquely identifies the entity instances. The key designation can be overridden in the network view level, so it functions mostly as a comment to network view creators.

### Network Relationship Types

Network relationship types map to local relationship types, making them visible in the network schema. The domain and range of the network relationship type must be network entity types defined in the network schema. These network entity types must map to the domain and range local entity types of the local relationship. A network relationship type must include one domain key attribute and one range key attribute which map to attributes of the domain and range local entity types, respectively. A network relationship type may also model zero or more of the other attributes of the local relationship type.

The information content of a network relationship type is the following:

```
network relationship <net domain> <net relation> <net range> is
    level <string> compartment <string> [ training ]
    source schema <local schema>
    source relationship <loc domain> <loc relation> <loc range>
    domain key <net attribute> from <loc attribute>
    range key <net attribute> from <loc attribute>
    [ <net attribute>
        from [ domain | range | relation ]+ <loc attribute> ]*
end network relationship
```

A local relationship type may be mapped into the network schema if and only if both the domain and range local entity types are already mapped to network entity types. The <net domain> must be a network entity type which maps to a local entity type named <l. domain>. The <net range> must be a network entity type which maps to a loca. ..tity type named <loc range>.

The rules for matching up <net domain> and <net range> with <loc domain> and <loc range> prevent name conflicts between local and network entity types. A local entity type must be modeled in the network schema before local relationships in which it participates are modeled so that the resulting network relationship can be expressed in terms of network entity types. Local entity type names are not visible in the network schema.

Zero or one attributes which map to domain entity attributes may be designated as the domain key. Zero or one attributes which map to range entity attributes may be designated as the range key. The domain and range key matching criteria of a network relationship type are the same as those of the local relationship type to which it maps. (More information pertaining to this topic is located in the subsection Matching Criteria.)

The type of each network attribute is the same as the type of the local attribute to which it maps. Attributes from the local relationship type may be renamed in the network relationship type. This handles the case where the domain and range entity types contain an attribute with the same name, but which contain different data. Renaming one or both attributes allows both to be visible in the network relationship type. The ATOMS ER language supports mapping attributes to the relationship (as opposed to the domain or range entities) however this capability is not currently implemented because none of the initial data servers support it.

## Network Views

The view is the ATOMS mechanism for automating the collection and presentation of data from distributed, multimedia sources. Views provide the context for all data retrieval in ATOMS.

Network views are named groups of entity and relationship types. A view is constructed by selecting one or more network entity and relationship types from one or more network subschemas. A (possibly complete) subset of the attributes of each entity and relationship type may be included in the view. In addition, new entity and relationship types can be defined which combine or model data in ways not present in the network schema. The most powerful entity and relationship type definition capabilities are present in the view construct, allowing the network user the maximum flexibility in selecting the data they wish to examine. A network view may be accessed by any user whose user name is on its access list.

The information content of a network view is the following:

```
network view <identifier> is
      access list [ <identifier> ]* ;
      level <string> compartment <string> [ training ]
      [ <view entity type>  |  <view relationship type>  ]*
      [ <query> ]*
end network view
```

When a network view is created, its access list contains only the name of the user who created it.

### View Entity Types

Views contain view entity types. View entity types allow network users to map to network subschema entity types making them visible in the view. View entity types also allow network users to combine data from multiple data sources and to specify multiple alternate sources which are to be consulted if the primary source is unavailable.

A view entity type consists of one or more consecutive sources, each of which may have zero or more alternate sources. A consecutive source is a mapping to a network entity type, which may include one or more of its attributes. A view entity type with multiple consecutive sources combines into one entity type instances of the same conceptual entity type that are stored in different data sources. This allows all the instances of a particular conceptual entity type to be retrieved, browsed, and searched without regard to where they are stored or how they are retrieved. By defining a view entity type, all the data on a network about a particular conceptual entity (for instance ships) can be searched using one operation.

A view entity type with multiple consecutive sources is assembled by retrieving each consecutive source in order and combining the resulting resulting instances into a continuous stream. This capability is conceptually different than connecting distinct entity types using a relationship type.

Each consecutive source of a view entity type may have zero or more alternate sources associated with it. An alternate source is a mapping to a network entity type which provides the same information as the consecutive source. Alternate sources allow ATOMS to make use of redundant or similar data when the consecutive source is unavailable. The alternate sources are consulted in a user defined order.

ATOMS does not require that all the consecutive and alternate sources of a view entity type provide the exact same attributes. This allows sources which are similar but not identical to be combined for searching, which would otherwise have to be searched individually. It also increases the flexibility of the alternate source mechanism because sources which provide only partial or similar data can be used in place of an unavailable source.

To successfully run a query on a view entity with multiple consecutive sources, it may be necessary to create special view entity types whose attributes all have the same name.

The information content of a view entity type is the following:

```
view entity <identifier> is
      level <string> compartment <string> [ training ]

      consecutive source <network entity name> is
            [ <attribute> [ key ] from <attribute> ]*
      end source

      [ [ alternate | consecutive ]+ source <network entity name> is
            [ <attribute> [ key ] from <attribute> ]*
      end source ]*
end view entity
```

The type of each view attribute is the same as the type of the network attribute to which it maps. The consecutive and alternate sources of a view entity type may map to network entity types defined in several different network subschemas. Any consecutive source may have multiple alternate sources. If a consecutive source is unavailable, then the first alternate source is consulted, if it is unavailable as well, then the second alternate source is consulted.

If a consecutive source and all its alternate sources are unavailable, then no entity instances are contributed to the view entity type by that consecutive source. The name of a view attribute must be unique within the consecutive or alternate source where it is defined.

The consecutive sources of a view entity type do not all need to define the same attributes. An alternate source of a view entity type does not need to define the same attributes as the consecutive source for which it is an alternate. The set of attributes of a view entity type is the union of all the attributes defined by the consecutive and alternate sources. If an attribute with the same name is defined by more than one source then its type must be the same in each definition, with the exception that string types of different lengths are permitted. (This rule is not currently enforced).

### View Relationship Types

Views contain view relationship type definitions. The view relationship type allows network users to relate network entity and view entity types. A relationship connecting view relationship types benefits from the consecutive and alternate source mechanism provided by view entity types.

The information content of a view relationship type is the following:

```
view relationship <domain> <relation> <range> is
    level <string> compartment <string> [ training ]

    domain key <attribute> from <attribute> <matching criteria>
    range key <attribute> from <attribute> <matching criteria>
    [ <attribute> from [ domain | range | relation ]+ <attribute> ]*
end view relationship
```

A view relationship type connects two view entity types. The type of each view attribute is the same as the type of the network attribute to which it maps.

A network entity type must be mapped into the same view as a view relationship type before it can be used as the domain or range of that view relationship type. A view entity type must be defined in the same view as a view relationship type before it can be used as the domain or range of that view relationship type.

The relation name of a view relationship type may be used in more than one mapped network or view relationship type. The mapped network entity types connected by a view relationship type may be defined in any network subschema. The domain and range view entity types may be the same (defines a recursive relationship). A recursive relationship may use the same attribute for the domain and range keys. This defines a relationship where each entity instance is related to itself (at least).

The ATOMS ER language supports mapping attributes to the relationship (as opposed to the domain or range), however this capability is not be implemented initially because none of the initial data servers support it.

## Queries

All retrieval in ATOMS is done in the context of a view. ATOMS allows users to retrieve entity and relationship data which matches a query. Each query applies to a single entity or relationship type. Queries are associated with entities and relationships and are stored along with the view in which they were created. This ensures that queries do not get separated from the context in which they are defined, and it allows them to be made visible on the network as part of a network view.

ATOMS queries are designed to be translated to SQL queries and to be implemented easily by non-SQL data servers. The information content of a query is the following:

```
query <identifier> is
     from [ entity <identifier> |
          relationship <domain> <relation> <range> ]+
     where  <expr>
     order [ <attribute> [ ascending | descending ]+ ]*
end query

<expr>     ::=    <term> [ <conj> <term> ]*
<term>     ::=    ( <query expression> )
<term>     ::=    <attribute> <op> <value>
<conj>     ::=    [ and | or ]+
<op>       ::=    [ "=" | "/=" | "<" | ">" | "<=" | ">=" | like ]+
<value>    ::=    [ <boolean> | <integer> | <float> | <string> ]+
```

Every attribute name specified in the where or order parts of the query must match an attribute name defined in the view entity or relationship type referenced in the from part.

## Network Correlation Sets

ATOMS provides the correlation mechanism to support the intelligence concepts of information correlation and fusion. A correlation allows an intelligence analyst to mark a set of one or more entities and/or relationships as correlated and then record a textual description of their significance. Correlations are stored for later retrieval and modification.

ATOMS allows a network user to create correlations between entity and relationship instances to which they have access. ATOMS provides the correlation set construct to support grouping of related correlations and to allow controlled sharing of correlations between network users. A correlation set is a named repository which may hold zero or more correlations. A user may create as many correlation sets as they wish. A network correlation set may be accessed by any user whose user name is on its access list.

The information content of a network correlation set is the following:

```
network correlation set <identifier>  is
     level <string> compartment <string> [ training ]
     access list [ <identifier> ]* ;
     [ <correlation> ]*
end network correlation set
```

The information content of a correlation is the following:

```
correlation <identifier> annotation <text> is
    level <string> compartment <string> [ training ]

    [<annotation> ] *

    [ member entity <identifier> is
        level <string> compartment <string> [ training ]
        [ <attribute> : <type> <value> ]#
    end member ]*

    [ member relationship <domain> <relation> <range> is
        level <string> compartment <string> [ training ]
        [ <attribute> : <type> <value> ]#
    end member ]*
end correlation
```

Correlations contain copies of the correlation data. Copying the correlated data and storing it in the correlation set provides faster retrieval and browsing of correlations, and prevents correlations from being lost due to schema changes. When a network correlation set is created, its access list contains only the name of the user who created it.

## Common Type Model

The same type structure is used throughout ATOMS from the local schema level to the view level. Communication to and from ATOMS data servers is in terms of the ATOMS type structure. Individual data servers are responsible for converting to and from the ATOMS type structure. Instead of spreading the problem throughout the various layers of ATOMS, the data conversion problem is isolated to the data server which in turn talks directly to the physical database and thus understands its types. This specific system design presents a more consistent model to the ATOMS database administrators and users.

The ATOMS common type model defines five data types:

```
boolean ( <yes (true) value>, <no (false) value> )
integer
float
string ( <integer length> )
file ( [ binary | text | tiff | xbm | dialogic ]+ )
```

These five data types employ the following rules.

1) ATOMS stores all values as strings and makes all data comparisons as string values. The data type provides additional information regarding the interpretation of the value.

2) The ATOMS integer type represents a 32 bit signed value.

3) The ATOMS float type represents an Ada "float" type value.

4) The ATOMS boolean type has three possible truth values: true, false, and indeterminate. A boolean type definition allows the specification of true and false constants. ATOMS compares retrieved boolean data with these values to determine the value of the boolean attribute. If the data does not match either the true or false value then the attribute is set to indeterminate. User definable truth values are required since there is no standard boolean representation among database managers, and different databases created with the same database manager may use different conventions.

5) The ATOMS string type represents fixed size string values from 1 to 255 characters in length.

## Network Users

A network schema contains the definitions of all the users known to ATOMS. The information content of a user definitions is the following:

```
[ user <identifier> is
      [ network_dba ]
      [ local_dba ]
      password <string>
      electronic_mail_address <string>
end user ]*
```

The name of an ATOMS user must be unique with respect to all other ATOMS users. ATOMS does not maintain a security clearance level and compartment list for users.

## Access Control in ATOMS

Unless otherwise specified in the following rules, no user may create, delete, retrieve, or modify any ATOMS local schema, network subschema, network view, network correlation set, the list of ATOMS users, or any elements they contain. The following access control rules are employed in ATOMS.

• When created, the access list of all local schemas, network subschemas, network views, and network correlation sets contains only the name of the user who created the element.

• The user with Network DBA privilege may retrieve and modify the list of ATOMS users. A user with Network DBA privilege may grant or revoke Network DBA and Local DBA privilege from any user (subject to the exception described in the next rule).

• Network DBA privilege may not be removed from a user if that user is only one with Network privilege. This rule prevents an ATOMS installation from losing access to functions requiring Network DBA privilege.

• A user with Network DBA privilege may create, delete, retrieve, and modify any local schema, network subschema, network view, or network correlation set in an ATOMS network.

• A user with Network DBA privilege, and only a user with Network DBA privilege, may modify the access list of any local schema, network subschema, network view, or network correlation set in an ATOMS network.

• A user with Local DBA privilege may create new local schemas.

• The user with Local DBA privilege may delete, retrieve, and modify any local schema which has their user name on its access list.

• Any user may retrieve any network subschema which has their user name on its access list.

• Any user may delete, retrieve, and modify any network view which has their user name on its access list.

• Any user may delete, retrieve, and modify any network correlation set which has their user name on its access list.

Since access rights are based on network subschemas and views rather than on entity types or relationship types, some thought must be given to efficiently detecting the problem of detecting relationships which jump out of the user's access world.

# ATOMS Data Servers

## UNIX Filesystem Data Server

The UNIX Filesystem Data Server allows UNIX files to be modeled as local entity types. Because CDROM data is mounted as a UNIX file system, CDROM disks should be modeled as local schemas which map to UNIX Filesystem Data Servers. The following rules were employed during system development.

One, the database node should be the network node name of the system where the files are located. This rule limits one local schema to modeling the files of one network node.

Two, the database name should be the path to the files to be modeled. Local entity types may specify further paths so the database name can be used as the base path of a collection of related files. May be the null string.

Three, the database type should be "unix_filesystem."

Four, the entity sources should be file specifications for UNIX files (which may include further directory paths, wildcards and extensions). The source is appended to the end of the database name to determine which files map to the local entity type.

Five, the UNIX Filesystem data server allows the retrieval of six specific entity type attributes:

| | | |
|---|---|---|
| name | string | The filename without the path. |
| full_name | string | The path name and filename. |
| size | integer | The file size in bytes. |
| date | string | The last modification date. |
| time | string | The last modification time. |
| content | file ( type ) | The contents of the file. |

Using any other attribute names or types will result in a query error.

## Network Schema Server

There is one Network Schema Server which stores the local schemas, network subschemas, network views, network correlation sets, and the ATOMS user list. The operation of the Network Schema Data Server is transparent to ATOMS users once it has been set up.

## Sybase and Oracle Data Servers

The Sybase and Oracle Data Servers handle accessing relational databases. There is one Sybase or Oracle Data Server process on each machine which contains a Sybase or Oracle Database. Local schemas can be written which target Sybase and Oracle databases. Retrieval of network entity and relationship types which map to local schemas accessing modeling relation databases are handled by Sybase or Oracle Data Servers.

## Dialogic Data Server

The Dialogic Data Server handles Dialogic message playback. Dialogic messages are stored as normal UNIX files, and can be modeled and accessed using the method described in the UNIX Filesystem Data Server. No local schemas need to be written which map directly to the Dialogic Data Server. This data server is used internally by ATOMS when the user requests playback of a message.

# ATOMS Development Process

The AAE project demonstrated a realistic approach to performing transparent access to heterogeneous, multimedia data. The following sections describe each key characteristic of the system.

## ATOMS User Interface

ATOMS uses a graphical, mouse-based, window-oriented interface which employes features such as pull-down menus, pop-up dialog and alert boxes, check boxes, option buttons and scrollable window panes.

The system exists inside the X Window System™. To open a window, the user clicks on the mouse button when the X Windows icon appears on the screen. Windows allow the user to manipulate schemas, subschemas, correlations, and view definitions. The different windows partition the display and provide a view of data to the users. A window may be moved or laid over one another so different windows can be viewed at the same time.

ATOMS is a user friendly system. Windows are composed of options and window panes. Users traverse panes by using the scroll arrows on the right side of the window panes. There are three types of window in ATOMS: main windows, selector windows and browser windows. Main windows and selector windows contain a schema graphic which is located on the right side of the window. The graphics provides a pictorial representation of ATOMS schema levels; they do not have any functional purpose inside ATOMS. Figure 4 is the local schema selector window; all selector windows in the system look similar to this one.

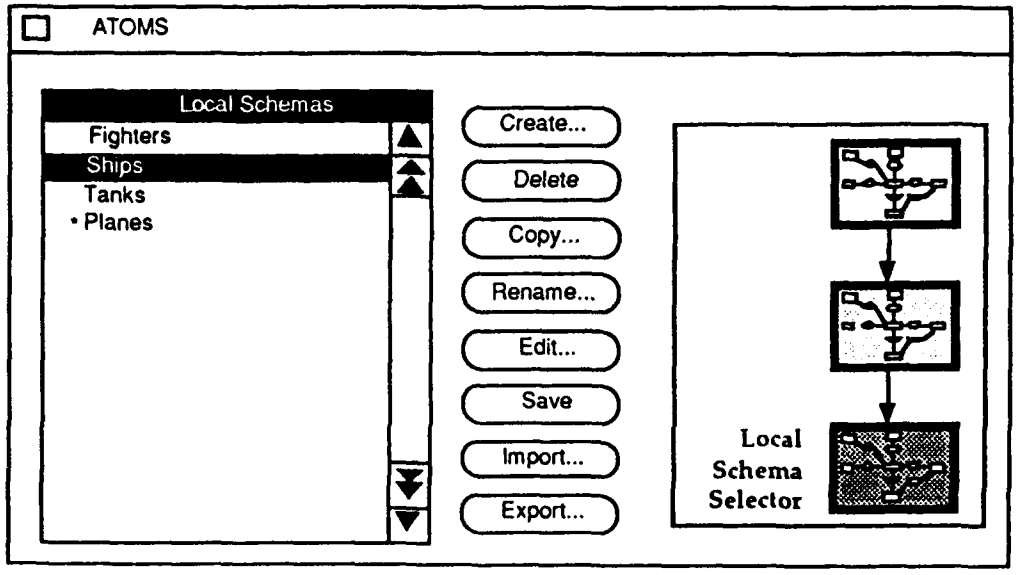


Figure 4. The Local Schema Selector Window

Users invoke options with option buttons or pull-down menus. Option buttons are located to the right of the window panes they manipulate. To activate an option, the user selects an item in the window pane and then selects an option button. All selections are done with the mouse. The user must select an item before selecting an option because ATOMS must know on which item to perform the operation, as shown in Figure 5. The only exceptions to this rule are the options create and import. The user simply selects these options to perform their operation. When either option is selected a dialog box comes to the window which requests the user enter a name. The dialog box provides a data field in which to enter the information.

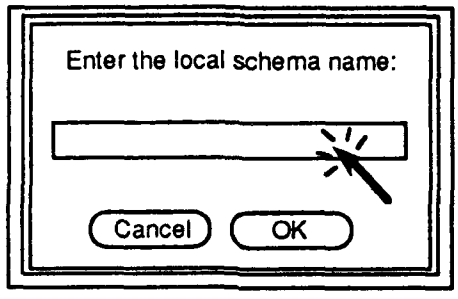


Figure 5. Click inside a data field before entering data.

To enter the information, the user must first click inside the data with the mouse. If the user starts typing before clicking inside the data field, the system beeps. When the user has entered the name, he or she must press the Return Key. Pressing the return key saves the information in the data field. These steps should be followed whenever entering data in data fields.

Pull-down menus are located at the top of all windows in the option bar. Each menu has its own title. Clicking on a menu title with the mouse causes the menu to pull down and reveal one or more options. To invoke an option in a menu, the mouse should be pulled down the menu in a continual motion until the desired option is selected.

ATOMS menus provide a sense of continuity throughout each schema level. Similiar menus appear in each browser window throughout the system so the user can easily go from one browser window to the next without readjusting to new menu options.

The ATOMS Main Menu appears in every ATOMS window. The menu in Figure 6 is the ATOMS Main Menu. The menu allows the user to set FTP setting or exit the system. Each browser window in ATOMS contains a Show Menu. This menu contains options which allow the user to decide what to display in the window. The browser windows also provide Edit Menus which provide options to modify, create or delete different data items.

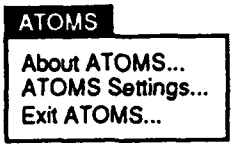


Figure 6. ATOMS Main Menu

Depending upon which schema level the user is working within, the browser window provides a menu which correlates to the particular level. For example, in the Local Schema Browser window this menu is Schema and in the Network Subschema Browser window this menu is Subschema. These menus provide options which allow the user to save any modifications and view security classification information. (The Local Schema Browser Schema Menu also provides a source option which allows the user to set up mapping to data sources.)

## Data Server and Local Schema Design Characteristics

Two key components of ATOMS are data servers and their local schemas; together they solve the problems of handling multimedia data and interfacing to heterogeneous databases.

Each type of data server is responsible for translating from its native database model and type structure into the ATOMS standard ER model and type structure. All data servers translate ATOMS ER queries to the native query language and they translate data in the native format to the ATOMS ER language types. This translation process has two benefits.

The first benefit is that the details of each database model are isolated to the data server which accesses that database type; hence, all schemas and types in ATOMS are homogeneous from the local schema level and up.

The second benefit of this translation process is that it allows new data servers to be written and incorporated into ATOMS without changing any of the existing system. The only criteria for incorporating a new data server into ATOMS is that its native database type maps into the ATOMS ER model and type structure. When this process is completed, the database is integrated into ATOMS.

An example integration of a data server could be one which is written to map local schemas to a commercial object base. The data server would take care of the mapping from the object base model and types to the ATOMS ER model and types. A new network server process could be created for the new data server and the object base information could be accessed by ATOMS without compiling or linking any ATOMS code.

Local Schemas present an interface where all data is modeled using the ATOMS ER language regardless of how it is physically stored by the data source. A great amount of research and design effort went into making local schemas rich enough to map many types of databases. Currently, ATOMS demonstrates mapping relational, voice and unix filesystems. The local schema definition is intended to be extensible to hierarchical, network, CODASYL and other database types. The Local DBA sets up the mapping between the local schema and the data source. This mapping is completed in the Local Schema Data Source Dialog Box shown in Figure 7.

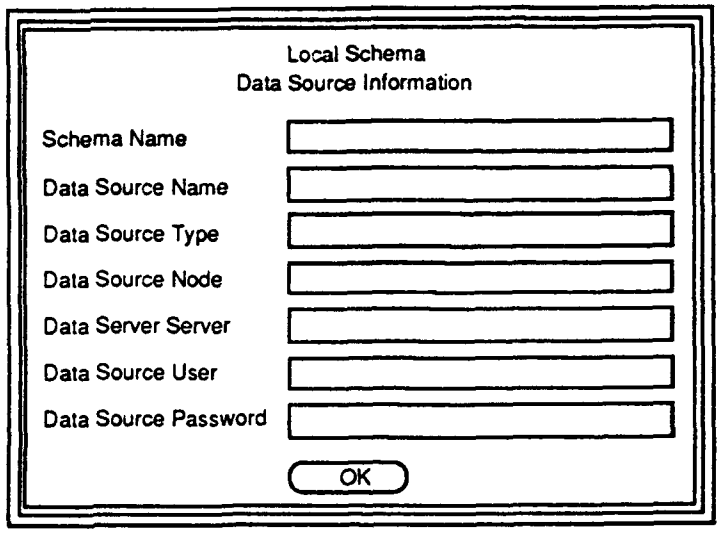


Figure 7. The ATOMS dialog box in which the user maps a local schema to a data source.

Each local schema in ATOMS describes one database of information stored by a single data server. All data servers communicate with ATOMS using the same network interface protocol, i.e., the ATOMS ER Language and query model. Because of this feature, the type of data server accessing the data source modeled by a local schema is not important to ATOMS.

One other important note concerning the design of the local schema is that some data servers may not implement the entire ER language or query model. ATOMS does not check if a local schema uses features not implemented by the data server to which it is tied.

Higher schema levels are much less reliant on the data model because they only look at the standard ATOMS ER model of the local schema. The local schemas hide the mapping to the data servers and only present the ATOMS names and types to the network subschema level. Only the data servers use the mapping information.

When a local schema is modeled, the network schema models the information in the local schemas in order to provide a single, unified schema for the user. This schema presents a single consistent interface to all the data available on the ATOMS

network. The network schema solves the problem of accessing distributed data. Separating the problems of accessing heterogeneous, multimedia databases in the local schema and accessing distributed data in the network schema simplifies the design and use of ATOMS.

At this time, the system does not support relationship types with attributes which belong solely to the relationship and not to the domain or range entities. A full scale system should analyze the data requirements needed to map each of the possible source database types to a local schema. A good validation of the mapping model would be to create example mappings to each of the possible source database types: relational, network, hierarchical, CODSAYL, ER, object-oriented, and any special purpose databases to be accessed by ATOMS. Data which pertains to accessing the database as whole must be captured, as in the local schema source window. Data which pertains to mapping to entities and relationships must also be captured. ATOMS does not presently capture information which would allow a relationship to be mapped directly to a data source object. The local schema mapping information must be complete in order to fully model a particular database type.

Changing the local schema mapping information during the system development or during system deployment will have a much greater ripple effect in the system code than adding a new type of data server which does not need any new mapping data.

## Security Classification in ATOMS

Although not intended to be a secure system, ATOMS does maintain and present Department of Defence (DoD) security classification information in order to identify classified material. ATOMS preserves the security level of classified material by displaying the security classification level and compartment level information in a security bar located at the very top of a window. From the lowest level to the highest level, the defined security classification levels in ATOMS are unclassified, confidential, secret, and top-secret. (Although not a true classification level, the term unclassified is part of the systems classification scale.) ATOMS uses this scale to enforce classification rules.

A local schema must be classified as high or higher than the highest local entity or relationship type which is contained by the local schema. The window, displayed in Figure 8, is a Local Schema Browser Window with sample data. The security bar displays the security classification of the local schema. This particular schema is TOP-SECRET and its compartment is S1.

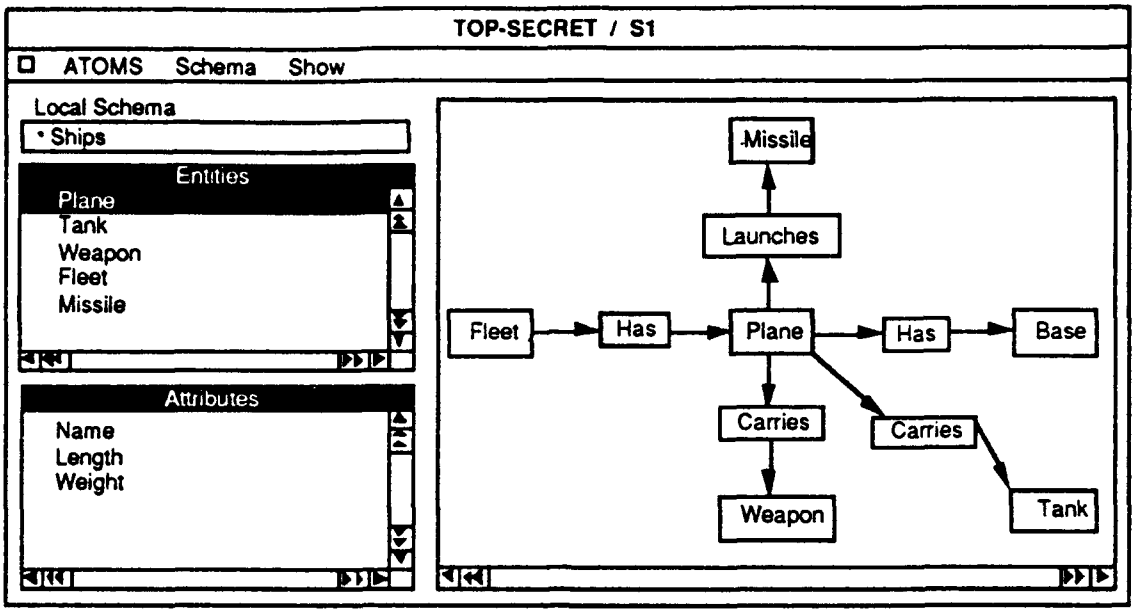


Figure 8. Example Local Schema Browser Window (classified for example purposes only.)

ATOMS allows information to be classified for training purposes. Any information which has been marked as classified for training purposes is indicated in a window's security bar with the legend "Classified for training purposes only." Schema elements which reference classified schema elements may not be classified for training purposes (this rule is not currently enforced.)

ATOMS allows the user to specify the security compartment of each construct except those which are marked as unclassified. The compartment is maintained in ATOMS, but no scale of compartment values is defined or enforced.

On the local schema level, the user must specify the security classification level of each local entity type. Classification information originates in the local entity types and flows upward through the levels of ATOMS schemas. Although they may be classified to a higher level, no ATOMS construct may be classified lower than the highest local entity type which the construct references. Local relationship types must be classified as high or higher than the higher of the classification levels of its domain and range entity types. Concerning compartments, if either or both the domain or range local entity types of a local relationship are marked with a security compartment, then the local relationship is automatically marked with the same compartment.

A user may also view the schema's security information by selecting a security option listed under the Schema Menu. Selecting this option opens the security dialog box listed in Figure 9.

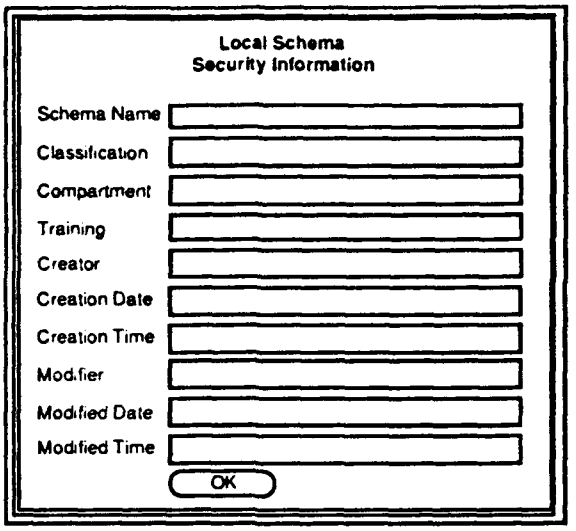


Figure 9. The Local Schema Security Dialog Box

Local schemas may contain elements which belong to one and only one compartment. If a schema contains at least one element which has a compartment, then the entire schema is marked with that compartment. Once a single element in a local schema is given a compartment, then all other elements in the schema can only be marked with that compartment or no compartment. Compartment mixing is not permitted at the local schema level.

The network subschema, network view and network correlation sets all follow the same the classification rules inside ATOMS. Each level is classified as high or higher than the highest entity or relationship type which is contained in the subschema, view or correlation set. The browser window for each of these levels is similar to the local schema browser window; the security bar at the top of the window displays the classification information. The user may also view this information in a security dialog box similar to the local schema security box. (Currently in ATOMS the user may lower the security information of a network view; this ability is a known error inside ATOMS.)

Concerning compartments, each different entity type must be marked with the same compartment as the entity type to which it maps; this rule applies to relationship types. Compartment mixing is not allowed on the network subschema level; however, a network view may contain elements which belong to more than one compartment and a network correlation set is marked with the union of all the compartments of the correlations contained in the correlation set.

The section, Future System Development Recommendations, specifies the known security and compartment rule violations within the system.

## Modification Log Files and Mail Messages

There are two methods which monitor and provide a record of modifications which occur in ATOMS: Log Files and mail messages.

ATOMS keeps a log file of all changes to local schemas, network schemas, network views, correlation sets, users, and access control lists. The network schema server process writes a time stamped message to a log file which is created when the network schema server is started. The log file names contain the time and date at which the network schema server was started. This format is employed to prevent new log files from writing over old ones.

The log files record time stamped messages which indicate all logins, including invalid attempts, log offs, schema changes, deletions, and renamings. It also records all ATOMS user changes and deletions and all access control list changes. ATOMS records the fact that a schema was changed but it does not record what change was made.

ATOMS automatically sends a mail message to every Network DBA whenever the Local DBA or Network User perform any actions inside the system. The mail messages allow the Network DBA to know exactly what the system users are doing inside ATOMS at all times and also provide a record of the modeling of system schemas, subschemas, views and correlations. Figure 10 is a sample mail message.

```
Message 24:
From aae Fri Aug 16 10:49:04  1991
Return-Path:  <aae>
Received: by aae.sps.com  (4.1/SMI-4.1)
        id AA00949; Fri, 16 Aug 91  10:49:02  EDT
Date: Fri, 16 Aug 91 10:49:02  EDT
From: aae (AAE Account)
Message-Id:  <9108161449.AA00949@aae.sps.com>
To: aae
Status: RO

Richard Smith deleted the correlation set
CORRELATION_PLANES.
```

Figure 10. Sample Mail Message

Mail messages are sent to a mail directory outside of ATOMS. When a Network DBA is created, he or she must be assigned an Electronic Mail . If they are not assigned an address, they will not receive any messages. Figure 11 is the ATOMS

User Editor Window. In this window, users are created, assigned DBA privileges, and given an electronic address if a Network DBA.

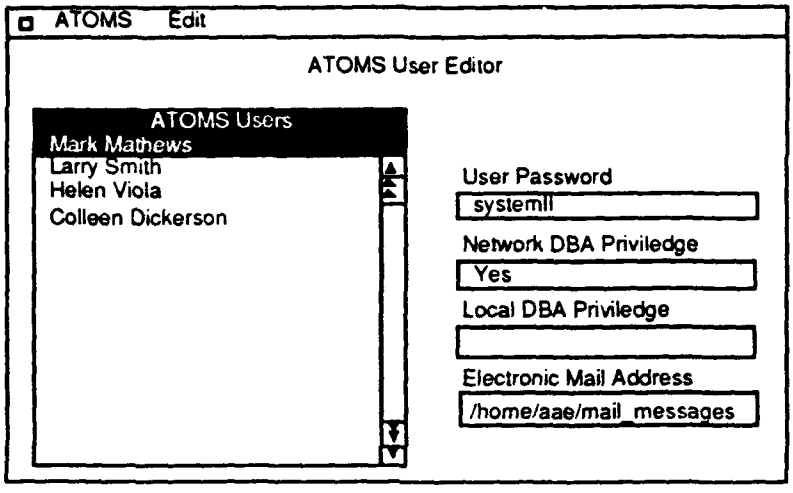


Figure 11. The Network DBA mail address is set in the User Editor Window.

We believe there potential for enhanced logging and schema version/rationale control; this topic is discussed more thoroughly in the section Future System Development Recommendations.

## User Defined Data Views and Queries

All data retrieval in ATOMS is done in the context of a view. Network Views contain a subset of the entity and relationship types in the network schema. A view may define new entity and relationship types. There are two powerful mechanisms for creating new entity types in views.

The first, known as the consecutive source, allows similar entity types to be joined consecutively into a single logical entity type. This source allows multiple entity types, possibly located in different database systems on different machines to be searched simultaneously using one query.

The second, known as the alternate source, allows an entity type to be substituted for a consecutive source if that source is unavailable. Multiple alternate sources may be defined for any sequential source. Relationships between view entity types inherit the power of the two mechanisms. The most powerful data modeling constructs are available at the application level, rather than being hidden in the Network Schema level.

Consecutive and alternate sources are created in the View Entity Editor Window, shown in Figure 12. Notice how this window displays the classification values of the network subschema and entity, to which the View maps, and of the consecutive

source selected in the Consecutive Sources window pane. (The options provided under the Show Menu allow the user to decide what data to display in the window; hence, the user could decide to view a consecutive source attribute's classification information or the user could decide to view the classification values of an alternate source or one of its attributes.)

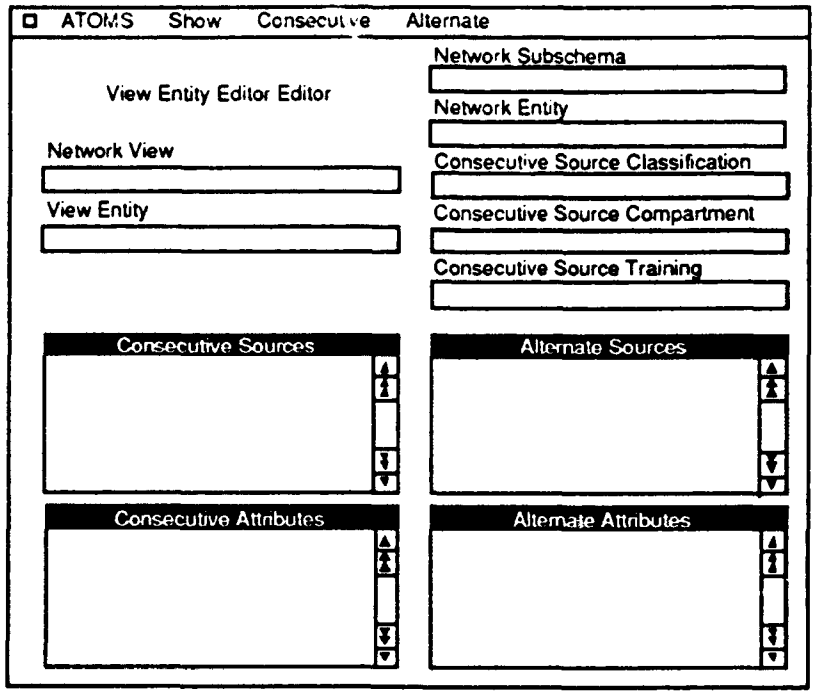


Figure 12. The View Entity Editor Window in which the user selects consecutive and alternate sources.

Figure 13 provides a conceptual view of how ATOMS stores a view entities' consecutive and alternate sources. The network user creates and executes queries

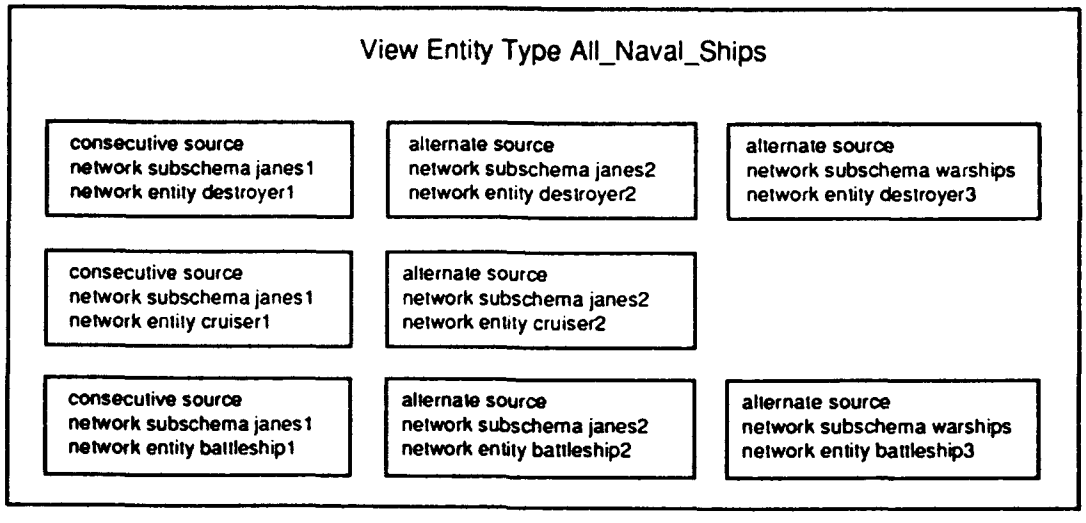


Figure 13 Conceptual View of Consecutive and Alternate Sources in ATOMS.

inside the View Browser Window. A query is created for each entity and each relationship in a view. Queries belong to entities and relationships like attributes belong to entities and relationships. This feature allows the user to exclusively tailor queries for particular data, one of the goals of ATOMS.

When a query is executed, ATOMS traverses the defined consecutive sources; however, there are two possible circumstances in which ATOMS switches to the alternate sources. One, the data server for a consecutive source is not running. Two, a mapping within the network view has been deleted.

When a query is complete, the Query Results Window opens and displays the retrieved entity instances. This window is shown in Figure 14. The retrieved attribute data is listed in two window panes. The first window pane lists all the retrieved values for the entities' or relationship's key attribute. One key attribute is listed per line. The order of data for each line in the pane is the name of the key attribute, the type of the key attribute and the value of the key attribute.

When the user selects one of the key attribute values, the other retrieved attributes for the entity or relationship appear in the bottom window pane. Each attribute is listed one per line in the order of attribute name, attribute type and attribute value. An attribute can be one of several types: string, float, integer, boolean, file(dialogic), file(tiff), file(xbm) and file(text).

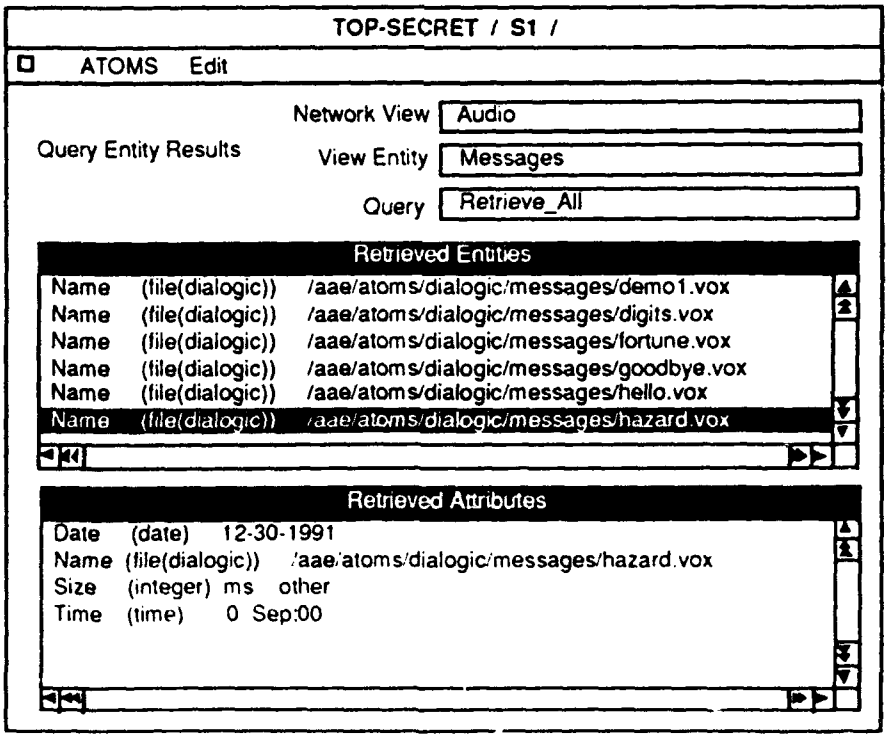


Figure 14. The Query Result Window.

The attribute types file(tiff) and file(xbm) hold graphics which can be selected and displayed in ATOMS. The attribute type file(dialogic) is a message attribute. The user can pick up the phone to hear the message which is stored in this attribute.

ATOMS provides Correlation Sets to support the intelligence concepts of information correlation and fusion. A Correlation Set allows a user to mark a set of one or more entities and/or relationships as correlated and then record a textual description of their significance. In ATOMS, Correlation Sets provide a storage medium so data can be easily retrieved and modified.

The Network DBA can create Correlation Sets for any data inside ATOMS. A Correlation Set is made up of Correlations which support a grouping of related entities and relationships and allow controlled sharing of correlations between system users. A user may create as many Correlation Sets as they wish. Correlation Sets can be accessed by any user whose user name is on its access list. When a set is created, its access list contains only the name of the user who created it. Figure 15 is an example Correlation Set Browser Window.

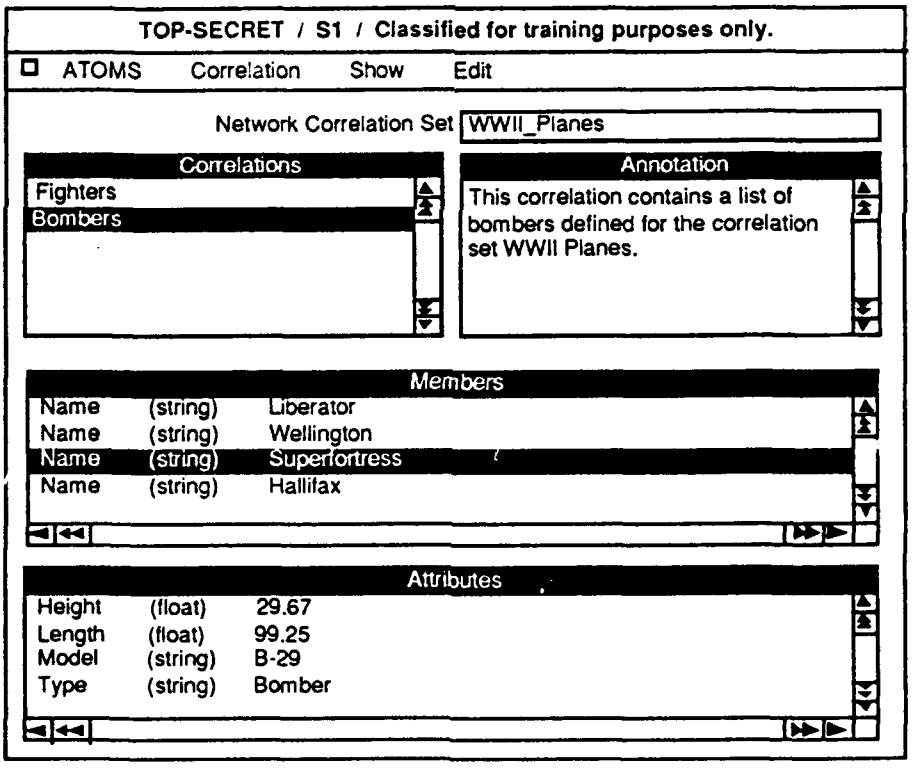


Figure 15. An Example Correlation Set Browser Window in which a user correlates entity and relationship instances.

In this window, the user can enter an annotation, in the annotation window pane, pertaining to a selected correlation. An annotation can be thought of as a place to enter key notes and information about the correlation.

# Future System Development Recommendations

This section discusses the lessons learned while researching and developing a system which provides transparent access to distributed, heterogeneous, multimedia data. It explains what we believe would be improvements to future system developments.

This section also specifies known system errors and explains how these errors could be rectified in future system developments.

## Security Verification Problems

ATOMS currently enforces the rule that no element can be classified lower than an element it references. This rule is enforced when the new element is created, but it is not enforced across schema levels if the lower level element is subsequently raised in classification., For instance, a network entity type which maps to a secret level local entity type must be classified as at least secret when the network entity type is created. This rule is enforced. Now if a local DBA goes back later on and raises the classification of the local entity type to TOP-SECRET the network entity type which maps to it is not raised.

ATOMS could try to search for all elements which reference an element when the classification of the element is raised. This however would require potentially searching all of the schemas in ATOMS each time a change was made to a lower level element. A better solution would be to create verification command for each of local schemas, network subschemas, network views, and correlation sets which would take each entity and relationship type and look at the classifications of the lower level elements to which they map and raise the entity or relationship types as needed. By going top down and on user demand instead of bottom up and automatically, the computational impact to the ATOMS systems is reduced to a manageable level. Such verification menu items could be implemented relatively easily.

## Multiple Compartments in ATOMS

The ATOMS definitions allow Network Views and Correlation Sets to contain information which is marked with more than one compartment. The current implementation does not always handle or enforce the compartment and classification rules correctly in these schema levels. This is largely due to the limitation on development time rather than the inherent complexity of the problem. The object oriented model set up and used by ATOMS can easily handle enforcing these rules, but the implementation was not completed.

Some additional analysis needs to be done to determine when it makes sense to allow multiple compartment classification, and how to control who can do it. More experience with ATOMS and an analysis of scenarios in which it could be used would be useful in determining exactly what should be done.

## TIFF Image Display

TIFF image display within ATOMS was not completely implemented. A program called TIFF_CONVERTER was written and is located on the SUN 3/80 in the /home/aae2/atoms_data/tiff/tiff_converter. This program translates TIFF images, which are very large files, into X Bitmap images. TIFF images can be incorporated into ATOMS by converting them to X bitmaps, which ATOMS does know how to display. The program can be run by entering the following command:

/home/aae2/atoms_data/tiff/tiff_converter <tiff filename>

When this command is executed, the system asks the user for the name of the TIFF file. The converted TIFF file is placed in a file called *CONVERTED.XBITMAP*.

## More Detailed Change Logging

An enhanced system could keep a more detailed log which records what changes are made to schemas and access control lists. The log file could explicitly record when entities, relationships, and attributes are created, deleted, or modified. It could also record a rationale for why changes have been made. The rationale information could also be stored with the schema itself to allow the history of the schema to be reviewed.

## Schema Configuration Management

ATOMS could provide a configuration management capability in which multiple versions of local schemas, network subschemas, network views, and network correlation sets could be maintained. The rationale and log information for the creation of each managed version would be stored. Old versions would be recovered if needed, and the complete history of why each version was created could be produced. This would be especially useful for recording the requirements which drove particular changes to be made, very important in an environment subject to multiple and possibly conflicting requirements.

## Numeric Types with Units

The ATOMS type model could be extended to include the use and understanding by ATOMS of standard units: feet, inches, pounds, dollars, etc. Local Schemas would record the units in which the fields of the database being modelled were expressed. Network Subschemas mapping to a local schema could specify different but compatible units if desired, and ATOMS would automatically make the necessary conversions. The Network View mappings to Network Subschemas could also specify similar unit conversions.

This capability is needed for two reasons. First, there is no way to tell the network users what units the numeric values displayed by ATOMS are expressed in, other than including this information in the name of the attribute, i.e. "LENGTH_IN_FEET". Second, there is no way to resolve the problem of two data sources which record information about the same logical entity type recorded in two different units. An example could be one data source recording airplane wingspans in feet and another recording wingspans in meters. If a network entity type used both databases as consecutive sources, half of the information about planes' wingspans' would come back in feet and the other half would come back in meters. This discrepancy would not be highlighted in any way by ATOMS and would be very confusing. A considerable amount of confusion could be avoided by adding units to ATOMS numeric types.

## Visual Display of Schemas

The schema diagrams displayed by the three schema browser windows use a rectangle to depict entities and relationships. It would be more clear visually if a different shape were used to depict one of them. A standard way to present ER diagrams would be to use diamonds or octagons to depict relationships.

Figure 16 below shows how a schema is currently represented in the system browser windows. (The window displayed comes on the screen when the user selects to browse the whole schema.) The schema representation in Figure 17 shows how the schema would appear if it followed the standard ER schema representation patterns we recommend above.

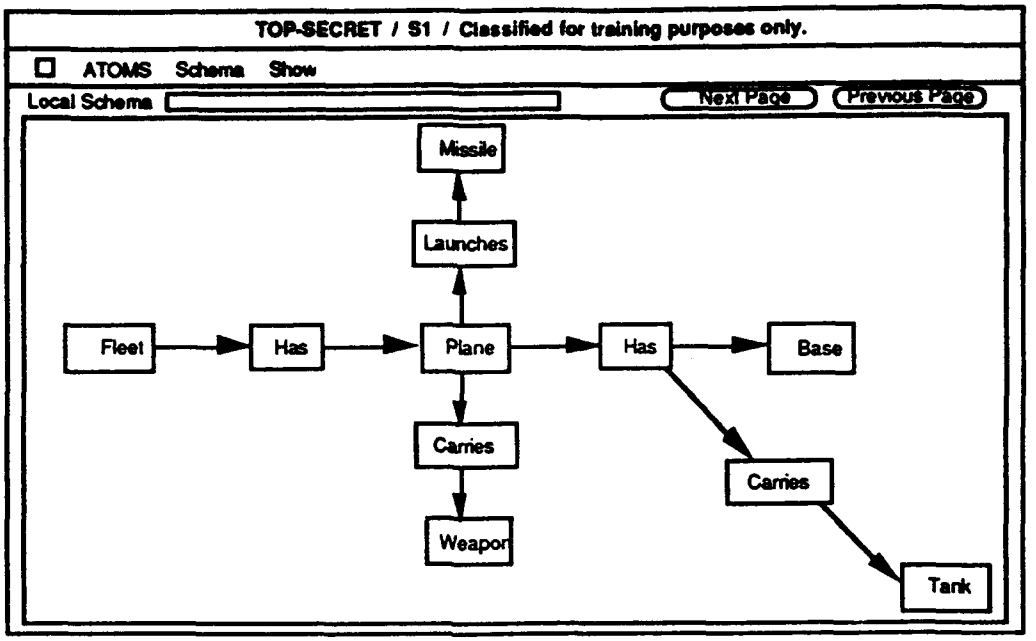


Figure 16. Current Schema Display Characteristics

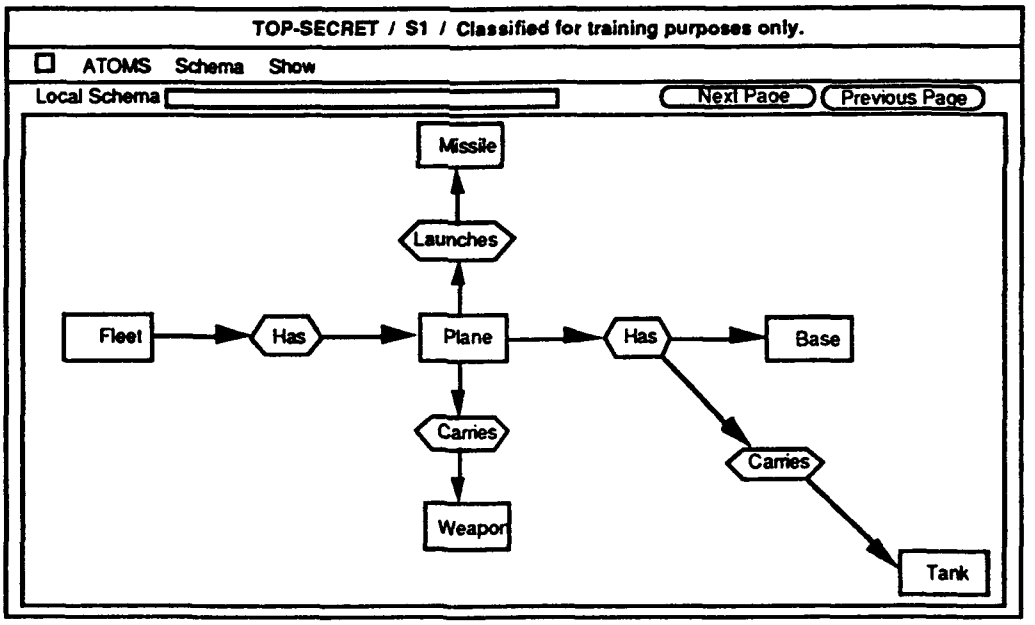


Figure 17. Suggested Improvements for Schema Display

## Stopping Automatic Relationship Deletion

When an entity type is deleted in ATOMS, all relationships are deleted which have the entity type as the domain or range. A dialog box appears on the screen to inform the user of each relationship that is being deleted. However, the user is not given the option to stop the deletion process. An improvement would be to inform the

user that deleting the entity type would cause relationships to be deleted and give them the option to cancel deleting the entity type.

## Querying Filesystem Data

The ATOMS File Server does not currently use the query constraints to determine which files to retrieve. It relies on the user entering the correct file specifications in the local schema database source and entity type source fields. A useful extension would be to allow filesystems to be queried in terms of the file name, size, date, and time attributes. For example, a query such as "SIZE > 100,000" currently has no affect. All files matching the database source and entity source are retrieved regardless of their size.

## Searching Instance Buffers

Sometimes queries in ATOMS will retrieve large numbers of instances. Rather than performing the query again to get a more restricted set, it would be useful to allow the retrieved instances to be searched. This would cut down on the amount of data which would need to be searched, as the user could progressively narrow down the list of matching instances, without going back to the entire database each time.

# Instructions on Running ATOMS and its Network Servers

This section contains instructions about how to run ATOMS hardware and its Network Servers. It also explains how servers can be set up to run a new machine.

## Starting ATOMS Hardware

The Sun 3/80 and the IBM 386 must both be started in order to run ATOMS. To start both machines, turn on all the peripheral power switches. Wait 15 seconds and then switch on the CPU power. Both machines automatically boot, checking their hardware and software, and then bringing up the login prompt. This process takes several minutes. When both machines have finished booting, start the ATOMS servers on the IBM 386 and the SUN 3/80

## Starting IBM 386 Data Servers

The IBM 386 computer, whose node name is "ike", runs the Oracle, Dialogic, and File Data Servers. To start these servers, follow these instructions: (Note: Attempting to start a server when it is already running produces an error message but does not interfere with the server.)

| | | |
|---|---|---|
| 1. | Log onto the IBM 386 | *user name:* atoms<br>*password:* _root_ |
| 2. | Move to the oracle directory | cd oracle |
| 3. | Start the oracle server | oracle_server < oracle_input & |
| 4. | Move to the dialogic directory | cd ../dialogic |
| 5. | Start the dialogic server | dialogic_server < dialogic_input & |
| 6. | Move to the file server directory | cd ../file_server |
| 7. | Start the ATOMS file server | atoms_file < atoms_input & |
| 8. | Move to the home directory | cd.. |

The user *must* remain logged onto the 386 to keep the servers running. Logging of will stop all three server processes, making them unavailable.

Entering the command "ps" produces a list of active processes belonging to a user. It can be used to determine whether or not the servers are running; names are

abbreviated to eight characters. The server processes run in the background, handling requests from multiple users. (Processes are queued in the order in which they were received)

Processes are killed in three particular instances: the user logs off the 386, the server process is killed by the user who started it or by the super user, or the server suffers a fatal error and dies (this instance is extremely unlikely.)

The user can kill a server process by entering the following command:

kill -9 <process id>

(The <process id> can be determined using the command "ps")

## Starting SUN 3/80 Data Servers

The SUN 3/80 runs the NSS and Sybase Servers. To start this server, the user must follow the instructions listed below.

| | | |
|---|---|---|
| 1. | Login to the SUN/30 | *user name:* root<br>*password:* aaeaae |
| 2. | Mount the CDROM drive | mount-a |
| 3. | Move to the sybase directory | cd /home/aae/sybase/install |
| 4. | Start the sybase database | startserver & |
| 5. | Log off | logout |
| 6. | Login to the SUN 3/80 | *user name:* aae<br>*password:* terinanike |
| 7. | Start X windows | startx |
| 8. | Move to the network schema server directory | cd atoms/net_server |
| 9. | Start the network schema server | nss_server < nss_input & |
| 10. | Move to the sybase server directory | cd../sybase_server |
| 11. | Start the sybase server | sybase_server < sybase_input & |

**Starting SUN 3/80 Data Servers (continued)**

| | | |
|---|---|---|
| 12. | Move to the file server directory | cd../file_server |
| 13. | Start the ATOMS file server | atoms_file_server < atoms_file_inputs & |
| 14. | Move to the home directory | cd../.. |

Once all ATOMS server are running and ATOMS itself can be started using the command run_atoms.

| Machine | Account | Password |
|---|---|---|
| SUN 3/80 (aae) | root | aaeaae |
| SUN 3/80 (aae) | aae | terinanike |
| SUN 3/80 (aae) | nss | terinanike |
| SUN 3/80 (aae) | analyst | <none> |
| SUN 3/80 | sybase | <none> |
| IBM 386 (ike) | root | _root_ |
| IBM 386 (ike) | atoms | _root_ |

## Setting Up New Servers

The oracle_input, dialogic_input, and atoms_input files contain the name which each server uses to listen for connections from ATOMS. These names are defined in the file /etc/services. This file is a standard UNIX file which lists the TCP port names and numbers for network services available on the machine. ATOMS uses one name for each of the server processes which can be running on a machine. The numbers must be between 2000 and 9999. Currently, numbers above 9000 are used.

The user is required to follow three steps to set up a server process on another machine:

**Setting Up New Servers (continued)**

| | |
|---|---|
| 1. | Add the server name and an unused port number to the /etc/services file on the new machine. |
| 2. | Copy the server name file and its input file to the new machine. |
| 3. | Start the server using the command: *server_name < server_input &* |

For example, to run the Oracle Server on a new machine, the user must add an Oracle Server name (for instance "oracleds") and an unused port number (such as "9001") to the /etc/services file on the new machine. Next, the user must copy the ORACLE_SERVER file and ORACLE_INPUT file (containing the correct name, i.e. "oracleds") to the new machine. The user starts the new server by entering the following command:

"oracler_server < oracle_input &"

Some UNIX versions require machine to be rebooted after the /etc/services file is changed in order for the changes to take effect. Once the server is running, local schemas which map to the new machine node and server name can be written and mapped up to network views and data on the new machine is ready for access.

One important note concerning the /etc/hosts file. On each machine running ATOMS, this file must contain entries for all of the machines which have ATOMS Data Servers running on them. This file is used by ATOMS networking software to get the TCP address of a machine given its name.

# AAE Program Studies and Metrics

The following subsections provide a detailed analysis of the design, code and testing of the Network Schema Server development, the ATOMS Main Program and Data Servers development, and the UIMS development. Summary statistics tables located at the end of this chapter contain the development effort information for each development phase. A cost metrics table is also provided.

## Network Schema Server Development

The high level design of the network schema server program was created by John Faure, and the implementation was primarily done by Sandra May, with some assistance and revision by John Faure. Sandra May, a software engineer at SPS, spent approximately 150 full time days between 11-9-90 and 6-25-91 implementing the network schema server program.

## ATOMS Main Program and Data Servers

The design and implementation of the ATOMS main program, Networking Software, ATOMS File Server, ORACLE server, Sybase Server, and Dialogic Server was performed by John Faure. Some of the lowest level sections of the network software and the database interfaces were created initially by Computer Science Innovations; subsequent revisions were performed by John Faure.

The User Interface Resource Files were created jointly by John Faure and MacKenzie Lovings, a software engineer and technical writer at SPS.

Using a resource based user interface approach saved a very large amount of engineering time and reduced the total number of lines of developed code significantly. These files can be thought of as a shorthand for creating and linking together user interface components such as windows, menus, and fields. We estimate that each line of code in a resource file is the equivalent of four to five lines of Classic-Ada code and perhaps ten lines of Ada code. These lines are included in the source code totals because they represented a significant development effort even though they saved a much greater amount of time.

John Faure spent 70 days from June 1990 to September 1990 completing the design of the Network Schema Server, the IRDS Subset, and rest of the ATOMS System. John Faure spent approximately 180 days on software implementation for the ATOMS System from October 1990 to August 1991 (the remaining 35 days were spent on vacation, business travel, and meetings). Having both the lead management and technical responsibility for the project impacted the amount of time which could be spent on pure engineering.

## User Interface Management System

Significant revisions and improvements to the SPS User Interface Management System (UIMS) were made by John Faure and Steve Nies in support of the AAE project. Most notably, the resource based approach was designed and implemented by John Faure to aid the ATOMS user interface development. This proved to be very successful, and illustrated an important lesson we learned during the development:

*Developing a mouse driven application with windows and menus requires an order of magnitude more effort than developing an equivalent application using a character based user interface.*

This is hardly a new finding, it has been reported in the literature several times in the last few years. Our object oriented and resource based user interface development approach is arguably the highest level fully general user interface development system available in Ada for The X Window System at the present time, and yet developing the resource files and associated Classic-Ada code required to animate the user interface took up more than 50% of the total time spent implementing the ATOMS main program. Our experience leads to the recommendation that no applications be developed for significant windowing systems without the use of high level construction tools, of which the typical X toolkits are barely acceptable, being at least a level below our own system in terms of development process support. The construction of mouse driven window based application is a significant undertaking, and should not be underestimated.

The UIMS used in the ATOMS represents the fourth complete redesign and implementation performed by SPS over the last three years. Each redesign has resulted in smaller, faster, and more functional code. The initial UIMS was at least twice the number of lines of code as the current one, and provided half the features. Resource file reading, multiple selection, bitmap graphics, and lines with arrow heads are some of the new features added to the UIMS by John Faure and Steve Nies.

## Program Statistic Metric Charts

This section presents the metric data which was collected on the ATOMS development. Raw collected data includes work product data, effort data, and cost data. Work product metrics attempt to characterize the amount of work performed and/or output generated in developing ATOMS. Work product metrics include:

1. Line of Code Counts, including:
    a. Ada source lines of code (Ada terminal semicolons)
    b. Lines of code (non-blank card images, including comments)

c. Non-comment lines of code (non blank card images, excluding comments)

d. UIMS Resource Language code (non-blank card images)

2. Design Information Counts, including:
   a. Number of Ada packages
   b Number of Classic-Ada classes (or "objects")
   c. Number of Classic-Ada methods
   d. Number of subprograms

3. Operational Capability Counts, including:
   a. Number of different user interface screens
   b. Number of user operations

Table 1 provides the raw data for the Line of Code (LOC) counts. Both detailed LOC count data by file and summary LOC count data by ATOMS subsystem are provided.

Table 2 provides the Design Information Counts by subsystem, and Table 3 provides the Operational Capability Counts.

Table 4 provides the raw effort data broken out by development activity as well as total labor costs. The data in Table 4 are taken from the AAE project Work Breakdown Structure (WBS) elements.

Table 5 provides summary productivity metrics by major activity based on the LOC counts, and Table 6 provides similar productivity metrics based on the Design Information Counts. Table 7 provides summary productivity metrics based on the Operation Capability Counts.

SPS is collecting similar data on other projects. Using multi-project data, we will be able to identify any across project trends that may be useful for estimating cost and effort on future projects. The Operational Capability based productivity metrics are especially attractive for cost estimating since the primary drivers (screens and operations) can often be accurately estimated early in a projects life cycle.

Table One. Line of Code (LOC) Count Data

| **Networking Files** | | |
|---|---|---|
| | | |
| Name | Total Lines | Semicolons |
| | | |
| CMI | 1382 | 375 |
| TCP | 1219 | 269 |
| TCP_Client | 374 | 70 |
| TCP_Server | 596 | 105 |
| **TOTALS** | **3571** | **819** |
| | | |
| | | |
| **Data Server Files** | | |
| | | |
| Name | Total Lines | Semicolons |
| | | |
| atoms_file_server.a | 689 | 259 |
| dia_ser.a | 191 | 81 |
| dldriver_s.a | 166 | 21 |
| dldriver_b.a | 521 | 108 |
| dlifc_s.a | 777 | 81 |
| dlifc_b.a | 631 | 116 |
| dltypes.a | 329 | 130 |
| log_spec.a | 351 | 153 |
| log_body.a | 441 | 195 |
| ora_int_s.a | 1055 | 163 |
| ora_ser.a | 285 | 116 |
| ora_sql_s.a | 60 | 13 |
| ora_sql_b.a | 455 | 127 |
| ora_typ_s.a | 223 | 35 |
| playback.a | 50 | 23 |
| sps_ftp_spec.a | 46 | 11 |
| sps_ftp_body.a | 178 | 63 |
| sybase_server.a | 281 | 116 |
| sybase_sql_spec.a | 61 | 13 |
| sybase_sql_body.a | 369 | 119 |
| unbounded_strings_spec.a | 109 | 19 |
| unbounded_strings_body.a | 222 | 67 |
| **TOTALS** | **7490** | **2029** |

# Table One. Line of Code (LOC) Count Data

| ATOMS Main Program Files--Classic Ada Source | | | |
|---|---|---|---|
| Name | Total Lines | Semicolons | Methods |
| atoms_copy_body.ca | 197 | 71 | |
| atoms_copy_spec.ca | 55 | 11 | |
| atoms_output_body.ca | 797 | 340 | |
| atoms_output_spec.ca | 73 | 20 | |
| atoms_paste_spec.ca | 56 | 12 | |
| atoms_schema_display_body.ca | 540 | 190 | |
| atoms_schema_display_spec.ca | 24 | 2 | |
| atoms_schema_manager_body.ca | 765 | 202 | |
| atoms_schema_manager_spec.ca | 175 | 39 | |
| atoms_schema_transfer_body.ca | 1196 | 429 | |
| atoms_schema_transfer_spec.ca | 124 | 35 | |
| atoms_session_body.ca | 602 | 156 | |
| atoms_session_spec.ca | 178 | 40 | |
| atoms_uims_body.ca | 227 | 81 | |
| atoms_uims_spec.ca | 72 | 16 | |
| atoms_user_body.ca | 689 | 242 | |
| atoms_user_spec.ca | 292 | 50 | |
| atoms_user_transfer_body.ca | 869 | 310 | |
| atoms_user_transfer_spec.ca | 92 | 21 | |
| audit_data_body.ca | 664 | 230 | |
| audit_data_spec.ca | 261 | 44 | |
| correlation_attribute_body.ca | 599 | 196 | |
| correlation_attribute_spec.ca | 234 | 43 | |
| correlation_body.ca | 719 | 263 | |
| correlation_set_body.ca | 69[illegible] | 223 | |
| correlation_set_spec.ca | 275 | 48 | |
| correlation_spec.ca | 264 | 45 | |
| cr_browser_command_body.ca | 945 | 278 | |
| cr_browser_command_spec.ca | 71 | 12 | |
| cr_selector_command_body.ca | 1027 | 306 | |
| cr_selector_command_spec.ca | 5[illegible] | 9 | |
| dialog_command_body.ca | 1104 | 419 | |
| dialog_command_spec.ca | 154 | 48 | |

Table One. Line of Code (LOC) Count Data

| | | | |
|---|---|---|---|
| ent_instance_body.ca | 718 | 240 | |
| ent_instance_spec.ca | 198 | 35 | |
| ent_query_results_body.ca | 497 | 155 | |
| ent_query_results_spec.ca | 235 | 38 | |
| instance_attribute_body.ca | 472 | 130 | |
| instance_attribute_spec.ca | 161 | 30 | |
| local_ent_att_body.ca | 509 | 170 | |
| local_ent_att_spec.ca | 232 | 41 | |
| local_entity_body.ca | 605 | 207 | |
| local_entity_spec.ca | 229 | 40 | |
| local_rel_att_body.ca | 699 | 237 | |
| local_rel_att_spec.ca | 289 | 50 | |
| local_relationship_body.ca | 1473 | 535 | |
| local_relationship_spec.ca | 454 | 86 | |
| local_schema_body.ca | 1222 | 455 | |
| local_schema_spec.ca | 383 | 67 | |
| ls_browser_command_body.ca | 2781 | 1041 | |
| ls_browser_command_spec.ca | 71 | 12 | |
| ls_export_body.ca | 531 | 313 | |
| ls_export_spec.ca | 61 | 12 | |
| ls_selector_command_body.ca | 1074 | 325 | |
| ls_selector_command_spec.ca | 55 | 9 | |
| network_attribute_body.ca | 559 | 194 | |
| network_attribute_spec.ca | 238 | 42 | |
| network_entity_body.ca | 650 | 226 | |
| network_entity_spec.ca | 243 | 42 | |
| network_relationship_body.ca | 1638 | 600 | |
| network_relationship_spec.ca | 510 | 94 | |
| network_subschema_body.ca | 911 | 320 | |
| network_subschema_spec.ca | 300 | 55 | |
| nss_operations_body.ca | 100 | 27 | |
| nss_operations_spec.ca | 46 | 11 | |
| nt_browser_command_body.ca | 3067 | 1058 | |
| nt_browser_command_spec.ca | 71 | 12 | |
| nt_selector_command_body.ca | 1230 | 370 | |
| nt_selector_command_spec.ca | 55 | 9 | |
| pick_command_body.ca | 1266 | 492 | |
| pick_command_spec.ca | 78 | 20 | |
| query_command_body.ca | 3262 | 1030 | |
| query_command_spec.ca | 57 | 10 | |

## Table One. Line of Code (LOC) Count Data

| | | | |
|---|---|---|---|
| rel_instance_body.ca | 891 | 305 | |
| rel_instance_spec.ca | 239 | 41 | |
| rel_query_results_body.ca | 681 | 221 | |
| rel_query_results_spec.ca | 320 | 50 | |
| security_command_body.ca | 620 | 252 | |
| security_command_spec.ca | 97 | 29 | |
| security_data_body.ca | 641 | 210 | |
| security_data_spec.ca | 228 | 44 | |
| sps_tiff_body.ca | 111 | 29 | |
| sps_tiff_spec.ca | 63 | 10 | |
| user_command_body.ca | 1431 | 468 | |
| user_command_spec.ca | 54 | 9 | |
| view_att_map_spec.ca | 26 | 3 | |
| view_body.ca | 914 | 321 | |
| view_ent_query_body.ca | 558 | 167 | |
| view_ent_query_spec.ca | 225 | 39 | |
| view_entity_body.ca | 767 | 283 | |
| view_entity_source_body.ca | 744 | 266 | |
| view_entity_source_spec.ca | 237 | 41 | |
| view_entity_spec.ca | 230 | 40 | |
| view_rel_query_body.ca | 896 | 287 | |
| view_rel_query_spec.ca | 341 | 57 | |
| view_relationship_body.ca | 1730 | 638 | |
| view_relationship_spec.ca | 517 | 95 | |
| view_spec.ca | 297 | 55 | |
| vw_browser_command_body.ca | 6007 | 2075 | |
| vw_browser_command_spec.ca | 71 | 12 | |
| vw_selector_command_body.ca | 1079 | 318 | |
| vw_selector_command_spec.ca | 55 | 9 | |
| **TOTALS** | **61386** | **19645** | **0** |

Table One. Line of Code (LOC) Count Data

| ATOMS Main Procedure--Ada Source Code | | |
|---|---|---|
| Name | Total Lines | Semicolons |
| atoms_debug_body.a | 116 | 35 |
| atoms_debug_spec.a | 64 | 11 |
| atoms_string_body.a | 268 | 71 |
| atoms_string_spec.a | 109 | 36 |
| generic_table_body.a | 224 | 67 |
| generic_table_spec.a | 111 | 26 |
| static_strings_body.a | 743 | 208 |
| static_strings_spec.a | 398 | 65 |
| string256.a | 19 | 2 |
| string32.a | 19 | 2 |
| string80.a | 19 | 2 |
| string80_map_spec.a | 25 | 4 |
| **TOTALS** | **2115** | **529** |

| IRDS Ada Source Code | | |
|---|---|---|
| Name | Total Lines | Semi-colons |
| btree_spec.a | 256 | 35 |
| btree_body.a | 1319 | 316 |
| irds_gen_doub_link_list_body.a | 598 | 185 |
| irds_gen_doub_link_list_spec.a | 272 | 38 |
| irds_gen_sto_man_body.a | 63 | 11 |
| irds_gen_sto_man_spec.a | 47 | 6 |
| irds_name_btree.a | 10 | 3 |
| irds_relationship_btree.a | 10 | 3 |
| nss_audit_trail_body.a | 206 | 57 |
| nss_audit_trail_spec.a | 54 | 7 |
| nss_diagnostics_body.a | 197 | 46 |
| nss_diagnostics_spec.a | 99 | 18 |
| nss_io_body.a | 145 | 44 |
| nss_io_spec.a | 53 | 10 |
| nss_listing_body.a | 479 | 161 |
| nss_listing_spec.a | 67 | 13 |
| nss_message_body.a | 209 | 56 |
| nss_message_spec.a | 79 | 10 |
| nss_scanner_body.a | 209 | 57 |

## Table One. Line of Code (LOC) Count Data

| | | |
|---|---|---|
| nss_scanner_spec.a | 23 | 3 |
| nss_spec.a | 25 | 3 |
| nss_token_table_body.a | 218 | 115 |
| nss_token_table_spec.a | 167 | 115 |
| nss_tokens.a | 28 | 11 |
| position_body.a | 106 | 25 |
| position_spec.a | 76 | 12 |
| underflow_sepr.a | 414 | 108 |
| **TOTALS** | **5429** | **1468** |

| **IRDS Classic-Ada Source Code** | | |
|---|---|---|
| | | |
| Name | Total Lines | Semicolons |
| | | |
| irds_attribute_body.ca | 86 | 17 |
| irds_attribute_boolean_body.ca | 90 | 18 |
| irds_attribute_boolean_spec.ca | 66 | 9 |
| irds_attribute_date_body.ca | 127 | 27 |
| irds_attribute_date_spec.ca | 69 | 10 |
| irds_attribute_integer_body.ca | 92 | 18 |
| irds_attribute_integer_spec.ca | 66 | 9 |
| irds_attribute_real_body.ca | 91 | 18 |
| irds_attribute_real_spec.ca | 66 | 9 |
| irds_attribute_spec.ca | 60 | 8 |
| irds_attribute_string_body.ca | 91 | 18 |
| irds_attribute_string_spec.ca | 67 | 10 |
| irds_attribute_time_body.ca | 128 | 27 |
| irds_attribute_time_spec.ca | 69 | 10 |
| irds_attribute_type_body.ca | 199 | 49 |
| irds_attribute_type_spec.ca | 145 | 19 |
| irds_base_body.ca | 568 | 195 |
| irds_base_spec.ca | 248 | 36 |
| irds_core_list_body.ca | 901 | 330 |
| irds_core_list_spec.ca | 203 | 25 |
| irds_entity_body.ca | 290 | 78 |
| irds_entity_spec.ca | 154 | 14 |
| irds_entity_type_body.ca | 375 | 83 |
| irds_entity_type_spec.ca | 245 | 23 |
| irds_list_body.ca | 640 | 220 |
| irds_list_spec.ca | 262 | 31 |

Table One. Line of Code (LOC) Count Data

| | | |
|---|---|---|
| irds_list_types.ca | 48 | 15 |
| irds_name_index_body.ca | 76 | 13 |
| irds_name_index_spec.ca | 78 | 12 |
| irds_object_body.ca | 587 | 128 |
| irds_object_spec.ca | 429 | 31 |
| irds_pers_list_body.ca | 1177 | 450 |
| irds_pers_list_spec.ca | 267 | 32 |
| irds_pers_list_types.ca | 92 | 22 |
| irds_relationship_body.ca | 216 | 57 |
| irds_relationship_index_body.ca | 73 | 12 |
| irds_relationship_index_spec.ca | 64 | 11 |
| irds_relationship_spec.ca | 136 | 15 |
| irds_relationship_type_body.ca | 613 | 155 |
| irds_relationship_type_spec.ca | 316 | 27 |
| irds_test.ca | 1457 | 746 |
| nss_ac_body.ca | 838 | 292 |
| nss_ac_spec.ca | 41 | 8 |
| nss_client.ca | 148 | 86 |
| nss_cr_body.ca | 3464 | 1196 |
| nss_cr_spec.ca | 91 | 23 |
| nss_create.ca | 302 | 122 |
| nss_delete_body.ca | 197 | 77 |
| nss_delete_spec.ca | 34 | 4 |
| nss_import_body.ca | 1321 | 417 |
| nss_import_spec.ca | 31 | 5 |
| nss_ls_body.ca | 3555 | 1220 |
| nss_ls_spec.ca | 112 | 23 |
| nss_nt_body.ca | 3228 | 1114 |
| nss_nt_spec.ca | 91 | 23 |
| nss_server.ca | 752 | 229 |
| nss_us_body.ca | 1097 | 418 |
| nss_us_spec.ca | 60 | 14 |
| nss_user_manager_body.ca | 575 | 211 |
| nss_user_manager_spec.ca | 102 | 30 |
| nss_vw_body.ca | 4036 | 1403 |
| nss_vw_spec.ca | 91 | 23 |
| **TOTAL** | **31193** | **9975** |

## Table One. Line of Code (LOC) Count Data

| UIMS | | |
|---|---|---|
| Name | Total Lines | Semi-colons |
| cartesian_body.ca | 108 | 27 |
| cartesian_spec.ca | 114 | 22 |
| keyboard_body.ca | 323 | 105 |
| keyboard_spec.ca | 149 | 18 |
| mouse_body.ca | 334 | 123 |
| mouse_spec.ca | 51 | 15 |
| resource.y | 2245 | 609 |
| resource_body.ca | 2069 | 634 |
| resource_fixup_body.ca | 502 | 145 |
| resource_fixup_spec.ca | 86 | 24 |
| resource_spec.ca | 37 | 7 |
| resource_tester.ca | 66 | 30 |
| sps_application_body.ca | 78 | 26 |
| sps_application_spec.ca | 91 | 14 |
| sps_arrow_body.ca | 313 | 92 |
| sps_arrow_spec.ca | 117 | 21 |
| sps_bitmap_body.ca | 362 | 81 |
| sps_bitmap_spec.ca | 102 | 17 |
| sps_boolean_body.ca | 743 | 291 |
| sps_boolean_rendering_body.ca | 262 | 93 |
| sps_boolean_rendering_spec.ca | 102 | 16 |
| sps_boolean_spec.ca | 490 | 64 |
| sps_check_box_body.ca | 234 | 64 |
| sps_check_box_spec.ca | 101 | 20 |
| sps_collection_body.ca | 46 | 9 |
| sps_collection_rendering_body.ca | 439 | 166 |
| sps_collection_rendering_spec.ca | 195 | 27 |
| sps_collection_spec.ca | 49 | 5 |
| sps_command_body.ca | 317 | 117 |
| sps_command_spec.ca | 223 | 38 |
| sps_composite_body.ca | 210 | 56 |
| sps_composite_spec.ca | 203 | 36 |
| sps_data_body.ca | 31 | 5 |
| sps_data_spec.ca | 44 | 5 |
| sps_desktop_body.ca | 318 | 126 |
| sps_desktop_spec.ca | 139 | 19 |
| sps_dictionary_body.ca | 458 | 161 |
| sps_dictionary_spec.ca | 326 | 55 |
| sps_dispatcher_body.ca | 445 | 148 |
| sps_dispatcher_spec.ca | 118 | 16 |
| sps_enumeration_rendering_body.c | 426 | 128 |
| sps_enumeration_rendering_spec.c | 115 | 20 |

Table One. Line of Code (LOC) Count Data

| | | |
|---|---|---|
| sps_float_body.ca | 374 | 143 |
| sps_float_rendering_body.ca | 178 | 65 |
| sps_float_rendering_spec.ca | 65 | 10 |
| sps_float_spec.ca | 317 | 44 |
| sps_graphic_body.ca | 1142 | 374 |
| sps_graphic_spec.ca | 580 | 92 |
| sps_host_info_body.ca | 245 | 79 |
| sps_host_info_spec.ca | 178 | 27 |
| sps_integer_body.ca | 356 | 132 |
| sps_integer_rendering_body.ca | 146 | 50 |
| sps_integer_rendering_spec.ca | 65 | 10 |
| sps_integer_spec.ca | 314 | 44 |
| sps_line_body.ca | 452 | 178 |
| sps_line_spec.ca | 142 | 23 |
| sps_list_body.ca | 1008 | 382 |
| sps_list_spec.ca | 295 | 47 |
| sps_main.ca | 109 | 89 |
| sps_modal_window_body.ca | 192 | 62 |
| sps_modal_window_spec.ca | 148 | 25 |
| sps_object_body.ca | 558 | 178 |
| sps_object_spec.ca | 285 | 31 |
| sps_persistent_boolean_body.ca | 71 | 17 |
| sps_persistent_boolean_spec.ca | 80 | 8 |
| sps_persistent_float_body.ca | 71 | 17 |
| sps_persistent_float_spec.ca | 80 | 8 |
| sps_persistent_integer_body.ca | 71 | 17 |
| sps_persistent_integer_spec.ca | 80 | 8 |
| sps_radio_button_body.ca | 198 | 58 |
| sps_radio_button_spec.ca | 101 | 20 |
| sps_rectangle_body.ca | 120 | 37 |
| sps_rectangle_spec.ca | 82 | 12 |
| sps_rendering_body.ca | 968 | 360 |
| sps_rendering_spec.ca | 251 | 44 |
| sps_string_body.ca | 1069 | 360 |
| sps_string_button_body.ca | 307 | 78 |
| sps_string_button_spec.ca | 119 | 21 |
| sps_string_rendering_body.ca | 63 | 16 |
| sps_string_rendering_spec.ca | 64 | 10 |
| sps_string_spec.ca | 550 | 88 |
| sps_tiff_body.ca | 111 | 29 |
| sps_tiff_spec.ca | 63 | 10 |
| sps_view_body.ca | 1852 | 602 |
| sps_view_spec.ca | 536 | 77 |
| sps_watch_body.ca | 109 | 37 |
| sps_watch_spec.ca | 35 | 4 |
| sps_window_body.ca | 1043 | 287 |

Table One. Line of Code (LOC) Count Data

| | | |
|---|---|---|
| sps_window_spec.ca | 292 | 47 |
| test_command_body.ca | 144 | 51 |
| test_command_spec.ca | 108 | 13 |
| test_sps_list.ca | 891 | 611 |
| test_sps_string.ca | 935 | 707 |
| test_uims.ca | 574 | 389 |
| **TOTAL** | **31368** | **9823** |
| | | |
| | | |
| **UIMS Ada Code** | | |
| | | |
| Name | Total Lines | Semi-colons |
| box_list.a | 5 | 3 |
| generic_doubly_linked_list_body.a | 611 | 185 |
| generic_doubly_linked_list_spec.a | 285 | 38 |
| generic_storage_manager_body.a | 91 | 11 |
| generic_storage_manager_spec.a | 76 | 6 |
| generic_table_body.a | 234 | 70 |
| generic_table_spec.a | 115 | 27 |
| object_list.a | 6 | 4 |
| position_body.a | 106 | 25 |
| position_spec.a | 76 | 12 |
| printer_body.a | 222 | 67 |
| printer_spec.a | 94 | 12 |
| resource_diagnostics_body.a | 197 | 46 |
| resource_diagnostics_spec.a | 99 | 18 |
| resource_goto.a | 1207 | 13 |
| resource_io_body.a | 145 | 44 |
| resource_io_spec.a | 50 | 10 |
| resource_listing_body.a | 494 | 168 |
| resource_listing_spec.a | 65 | 13 |
| resource_scanner_body.a | 236 | 64 |
| resource_scanner_spec.a | 23 | 3 |
| resource_shift_reduce.a | 1274 | 10 |
| resource_tokens.a | 66 | 11 |
| view_info.a | 19 | 10 |
| window_list.a | 17 | 9 |
| **TOTAL** | **5813** | **879** |

Table One. Line of Code (LOC) Count Data

| ATOMS Total Lines and Non-Blank Lines | | | |
|---|---|---|---|
| Source Code | Total-Lines | Non-Blank/ Non-Comment Lines | Semi-colons |
| Networking Files | 3571 | 1197 | 819 |
| Data Server Files | 7490 | 4705 | 2029 |
| ATOMS Classic-Ada Source | 61386 | 43517 | 19645 |
| ATOMS Ada Source | 2115 | 1166 | 529 |
| IRDS Classic-Ada Source | 31193 | 28105 | 9975 |
| IRDS Ada Source | 5429 | 4247 | 1468 |
| UIMS Classic-Ada Source | 31664 | 21817 | 9823 |
| UIMS Ada Source | 5517 | 5295 | 879 |
| Resource Files | 36183 | 32450 | N/A |
| **TOTALS** | **184548** | **142499** | **45167** |

Table Two. Design Information Count Data

| ATOMS Classes, Methods, Subprograms, and Packages | | | | |
|---|---|---|---|---|
| Source Code | Classes | Subprograms | Methods | Packages |
| Networking Files | N/A | 69 | N/A | 4 |
| Data Server Files | N/A | N/A | 2705 | 2 |
| ATOMS Classic-Ada Source | 106 | 142 | 964 | 12 |
| ATOMS Ada Source | N/A | 133 | N/A | 12 |
| IRDS Classic-Ada Source | 130 | 155 | 746 | 22 |
| IRDS Ada Source | N/A | 207 | N/A | 28 |
| UIMS Classic-Ada Source | 103 | 115 | 636 | 8 |
| UIMS Ada Source | N/A | 234 | N/A | 34 |
| Resource Files | N/A | N/A | N/A | N/A |
| **TOTALS** | **339** | **1055** | **5051** | **122** |

Table Three. Operation Capability Count Data

| Windows | Number of Each Window | Number of Options |
|---|---|---|
| ATOMS Login Window | 1 | 2 |
| Selector Windows | 4 | 52 |
| Browser Windows | 4 | 126 |
| User Editor Window | 1 | 7 |
| Access Control Window | 1 | 12 |
| Main Windows | 3 | 16 |
| View Entity Editor Window | 1 | 18 |
| Query Editor Window | 1 | 11 |
| Query Result Window | 1 | 12 |
| System Dialog Boxes | 22 | 42 |
| **TOTAL** | **41** | **298** |

## Table Four. Effort and Cost WBS Data

| Activity | Labor Hours | Labor Cost |
|---|---|---|
| | | |
| Management | 653 | |
| | | |
| Requirements | | |
| Requirements Definition | 201 | |
| Operational Concept | 258 | |
| Software Req. Specification | 425 | |
| Testbed Configuration | 190 | |
| User Interface Prototype | 180 | |
| | | |
| Design | | |
| IRDS Research & Design | 627 | |
| ATOMS Design | 3,223 | |
| | | |
| Code and Test | | |
| Object Base | 587 | |
| UIMS | 759 | |
| Btree and Sets | 1,336 | |
| IRDS and NSS | 773 | |
| ATOMS | 1,360 | |
| UIMS Resource Files | 160 | |
| Data Servers | 2,438 | |
| | | |
| Design/Code/Test SUBTOTAL | 11,262 | |
| | | |
| GRAND TOTAL | 23,778 | $816,975 |

Table Five. Productivity Data

| Major Activity | Total-Lines/ Hour | Non-Comment Lines/Hour | Semi-Colons/Hour |
|---|---|---|---|
| | | | |
| Total Project (all phases) | 7.76 | 5.99 | 1.90 |
| | | | |
| Design, Code, Test Phases | 16.39 | 12.65 | 4.01 |
| | | | |
| | | | |
| | $/Total Line | $/Non-Comment Line | $/Semi-Colon |
| | | | |
| Total Project (all phases) | $4.43 | $5.73 | $18.09 |

## Table Six. Design Information Productivity Data

| Major Activity | Hours/ Class | Hours/ Subprogram | Hours/ Method | Hours/ Package | |
|---|---|---|---|---|---|
| | | | | | |
| Total Project (all phases) | 70.14 | 22.54 | 4.71 | 194.91 | |
| | | | | | |
| Design, Code, Test Phases | 33.22 | 10.68 | 2.23 | 92.32 | |
| | | | | | |
| | | | | | |
| | $/Class | $ / Subprogram | $/ Method | $/Package | |
| | | | | | |
| Total Project (all phases) | $2,410 | $774 | $162 | $6,697 | |

Table Seven. Operational Capability Productivity Data

| Major Activity | Hours/ Window | Hours/ Operation |
|---|---|---|
| | | |
| Total Project (all phases) | 580 | 80 |
| | | |
| Design, Code, Test Phases | 275 | 38 |
| | | |
| | | |
| | $/Window | $ / Operation |
| | | |
| Total Project (all phases) | $19,926 | $2,742 |

# Test Data

Test databases were created to illustrate each of the four types of data servers and each of the types of multimedia data supported by ATOMS. Local schemas, network schemas, and network views were created which access each of the test databases. These schemas and be edited and examined for an example of how to model each of the types of data supported by ATOMS. There are many different ways any database can be modeled using local schemas, and many ways in which the local schemas can be combined into network schemas and views. So as not to be confusing, the test schemas illustrate a very straightforward modeling and combination of the test databases.

## ORACLE Test Data

The ORACLE test data follows. (The data for this table is located in the Oracle Data directory.)

```
DATABASE NAME = AAE
DATABASE USER = AAE
DATABASE PSWD = AAE

TABLE PLANES (
        MODEL                   CHAR(6),
        NAME                    CHAR(15),
        TYPE                    CHAR(7),
        SUBTYPE                 CHAR(7),
        NATION                  CHAR(5),
        MANUFACTURER            CHAR(17),
        WINGSPAN                NUMBER,
        LENGTH                  NUMBER,
        HEIGHT                  NUMBER,
        CREW_SIZE               NUMBER,
        MAX_SPEED               NUMBER,
        CEILING                 NUMBER,
        RANGE                   NUMBER,
        ENGINE_MAKE             CHAR(20),
        ENGINE_NUMBER           NUMBER,
        ENGINE_HP               NUMBER,
        BOMBLOAD                NUMBER,
        CANNON_TYPE             CHAR(2),
        CANNON_NUMBER           NUMBER,
        MACHINE_GUN_CALIBRE     CHAR(3),
        MACHINE_GUN_NUMBER      NUMBER,
        PICTURE_AVAIL           CHAR(1))
```

## SYBASE Test Data

The SYBASE Test Data follows.

DATABASE NAME = wwii
DATABASE USER = aae
DATABASE PSWD = aae

Sybase is case sensitive with respect to user, database, table and field names. When modeling a Sybase database in ATOMS, enter all the database and source information in the exact case in which it was created. All the Sybase test data was created using upper case names.

```
TABLE ARTILLERY (
        NATION                  CHAR(5),
        TYPE                    CHAR(14),
        CALIBRE                 NUMBER,
        RANGE                   NUMBER,
        MIN_ELEVATION           NUMBER,
        MAX_ELEVATION           NUMBER,
        TRAVERSE                NUMBER,
        WEIGHT                  NUMBER,
        MUZZLE_VELOCITY         NUMBER,
        RATE_OF_FIRE                    NUMBER,
        PENETRATION_THICKNESS   NUMBER,
        PENETRATION_ANGLE       NUMBER,
        PENETRATION_DISTANCE    NUMBER,
        TIME_TO_EMPLACE         NUMBER,
        PICTURE_AVAIL           CHAR(1))

TABLE FLEET_SIZES (
        COUNTRY                 CHAR(32),
        BATTLESHIPS             NUMBER,
        BATTLECRUISERS          NUMBER,
        SEAPLANE_CARRIERS       NUMBER,
        AIRCRAFT_CARRIERS       NUMBER,
        HEAVY_CRUISERS          NUMBER,
        LIGHT_CRUISERS          NUMBER,
        DESTROYERS              NUMBER,
        ESCORTS                 NUMBER,
        SUBMARINES              NUMBER)
```

```
TABLE GERMAN_RUSSIAN_FRONT (
        ARMY_NAME                CHAR(32),
        ARMY_GROUP               CHAR(32),
        COMMANDER                CHAR(32),
        PANZER_DIVISIONS         NUMBER,
        INFANTRY_DIVISIONS       NUMBER)

TABLE GERMAN_TANKS (
        MODEL                    CHAR(32),
        TYPE                     CHAR(60),
        WEIGHT_TONS              NUMBER,
        LENGTH_FEET              NUMBER,
        WIDTH_FEET               NUMBER,
        HEIGHT_FEET              NUMBER,
        ROAD_SPEED_MPH           NUMBER,
        TERRAIN_SPEED_MPH        NUMBER,
        FRONT_ARMOR_MM           NUMBER,
        SIDE_ARMOR_MM            NUMBER,
        CREW_MEMBERS             NUMBER)

TABLE MARINES (
        SHIP            CHAR(20),
        COMMANDER       CHAR(10),
        COMPLEMENT      NUMBER,
        NOTES           CHAR(40))

TABLE PLANES (
        MODEL                    CHAR(6),
        NAME                     CHAR(15),
        TYPE                     CHAR(7),
        SUBTYPE                  CHAR(7),
        NATION                   CHAR(5),
        MANUFACTURER             CHAR(17),
        WINGSPAN                 NUMBER,
        LENGTH                   NUMBER,
        HEIGHT                   NUMBER)

TABLE SHIPS (
        NAME            CHAR(20),
        NATION          CHAR(10),
        TYPE            CHAR(20),
        ENGINES         CHAR(5),
        LENGTH          NUMBER,
        BEAM            NUMBER,
        DISPLACEMENT    NUMBER)
```

The following data was entered for the Artillery table:

```
CREATE TABLE ARTILLERY (
 NATION              char(5),
 TYPE               char(14),
 CALIBRE             float,
 RANGE               float,
 MIN_ELEVATION       float,
 MAX_ELEVATION       float,
 TRAVERSE            float,
 WEIGHT              float,
 MUZZLE_VELOCITY     float,
 RATE_OF_FIRE        float,
 PENETRATION_THICKNESS  float,
 PENETRATION_ANGLE     float,
 PENETRATION_DISTANCE  float,
 TIME_TO_EMPLACE     float,
 PICTURE_AVAIL       char(1))
go
insert ARTILLERY
VALUES ("GER  ","Anti-tank     ",50,3000,-
13,22,60,2016,3280,16,71,30,60,0,"N")
go
insert ARTILLERY
VALUES ("GER  ","Anti-tank     ",75,8750,-
5,22,60,3350,3070,15,3.43,30,1000,0,"N")
go
insert ARTILLERY
VALUES ("USA  ","Howitzer      ",75,9600,-5,45,6,1442,1250,6,0,0,0,0,"N")
go
insert ARTILLERY
VALUES ("USA  ","Anti-aircraft ",90,33800,-
10,80,360,28000,2700,30,7,0,0,0,"N")
go
insert ARTILLERY
VALUES ("USA  ","Anti-tank     ",37,13000,-
10,15,60,912,2640,25,2.4,20,520,0,"N")
go
insert ARTILLERY
VALUES ("GB   ","Howitzer      ",87.6,13400,-
5,40,8,7335,1485,5,2.52,30,1000,0,"N")
go
insert ARTILLERY
VALUES ("GB   ","Anti-tank     ",76.2,11500,-
6,16,60,6700,3100,20,9.13,1100,0,0,"N")
go
insert ARTILLERY
VALUES ("GB   ","Howitzer      ",140,17000,-
5,45,60,12850,1675,2,0,0,0,0,"N")
go
insert ARTILLERY
VALUES ("GER  ","Anti-aircraft ",20,12465,-
10,100,360,2979,2959,800,0,0,0,0,"N")
```

```
go
insert ARTILLERY
VALUES ("USA  ","Howitzer      ",155,25395,-
1,63,60,26600,2800,1,30,0,0,0,"N")
go
insert ARTILLERY
VALUES ("USA  ","Howitzer      ",203.2,18510,-
2,65,60,25000,1950,1,45,0,0,0,"N")
go
insert ARTILLERY
VALUES ("USA  ","Anti-tank     ",81,16100,-
5,30,45,4785,2800,20,0,0,0,0,"N")
go
insert ARTILLERY
VALUES ("USA  ","Anti-aircraft ",120,47400,-
5,80,360,43600,3100,12,40,0,0,0,"N")
go
insert ARTILLERY
VALUES ("USA  ","Self-propelled",155,18700,-
5,30,28,51800,2410,4,0,0,0,0,"N")
go
insert ARTILLERY
VALUES ("USA  ","Self-propelled",105,12200,-
5,32,25,46400,1550,4,0,0,0,0,"N")
go
```

The following data was entered for the fleet sizes table:

```
CREATE TABLE FLEET_SIZES (
 COUNTRY          char(32),
 BATTLESHIPS      float,
 BATTLECRUISERS   float,
 SEAPLANE_CARRIERS float,
 AIRCRAFT_CARRIERS float,
 HEAVY_CRUISERS   float,
 LIGHT_CRUISERS   float,
 DESTROYERS       float,
 ESCORTS          float,
 SUBMARINES       float)
go
insert FLEET_SIZES values
("Germany",  2, 2, 0 ,0, 4,  5, 17,  8, 57 )
go
insert FLEET_SIZES values
("France",   6, 0, 1, 1, 7, 12, 70, 0, 77)
go
insert FLEET_SIZES values
("Britain", 12, 3, 2, 7, 15, 47, 159, 38, 38)
go
insert FLEET_SIZES values
("Italy",    6, 0, 1, 0, 7, 12, 61, 83, 98)
go
```

```
insert FLEET_SIZES values
("USSR",    3, 0, 0, 0, 1, 9, 59, 0, 0)
go
insert FLEET_SIZES values
("Japan",   10, 0, 3, 10, 18, 18, 113, 0, 63)
go
insert FLEET_SIZES values
("USA", 17, 0, 0, 7, 18, 19, 171, 0, 112)
go
```

The following data was entered for the German Russian Front

```
CREATE TABLE GERMAN_RUSSIAN_FRONT (
 ARMY_NAME          char(32),
 ARMY_GROUP         char(32),
 COMMANDER          char(32),
 PANZER_DIVISIONS   float,
 INFANTRY_DIVISIONS float)
go
insert GERMAN_RUSSIAN_FRONT values
("Eighteenth Army", "North", "von Kuchler",0, 8)
go
insert GERMAN_RUSSIAN_FRONT values
("Fourth Panzer Group", "North", "Hoeppner", 3, 6)
go
insert GERMAN_RUSSIAN_FRONT values
("Sixteenth Army", "North", "Busch", 0, 12 )
go
insert GERMAN_RUSSIAN_FRONT values
("(Reserve)", "North", "", 0, 1)
go
insert GERMAN_RUSSIAN_FRONT values
("Ninth Army", "Centre", "Strauss", 0, 9)
go
insert GERMAN_RUSSIAN_FRONT values
("Third Panzer Group", "Centre", "Hoth", 4, 7)
go
insert GERMAN_RUSSIAN_FRONT values
("Fourth Army", "Centre", "von Kluge", 0, 16)
go
insert GERMAN_RUSSIAN_FRONT values
("Second Panzer Group", "Centre", "Guderian", 5, 9)
go
insert GERMAN_RUSSIAN_FRONT values
("(Reserve)", "Centre", "", 0, 1)
go
insert GERMAN_RUSSIAN_FRONT values
("Sixth Army", "South", "von Reichenau", 0, 6)
go
insert GERMAN_RUSSIAN_FRONT values
("First Panzer Group", "South", "von Kleist", 5, 9)
go
insert GERMAN_RUSSIAN_FRONT values
```

```
("Seventeenth Army", "South", "Stulpnagel", 0, 13)
go
insert GERMAN_RUSSIAN_FRONT values
("Eleventh Army", "South", "Schobert", 0, 7)
go
insert GERMAN_RUSSIAN_FRONT values
("Rumanian Army", "South", "Antonesch", 0, 14)
go
insert GERMAN_RUSSIAN_FRONT values
("(Reserve)", "South", "", 0, 3 )
go
insert GERMAN_RUSSIAN_FRONT values
("OKH", "Army High Command", "Brauchitsch", 2, 22)
go
```

The following data was entered for the table German Tanks.

```
CREATE TABLE GERMAN_TANKS (
 MODEL          char(32),
 TYPE           char(60),
 WEIGHT_TONS    float,
 LENGTH_FEET    float,
 WIDTH_FEET     float,
 HEIGHT_FEET    float,
 ROAD_SPEED_MPH  float,
 TERRAIN_SPEED_MPH float,
 FRONT_ARMOR_MM  float,
 SIDE_ARMOR_MM   float,
 CREW_MEMBERS    float)
go

insert GERMAN_TANKS values
("Pz. Kpfw. I", "Light Tank", 5.7, 13.2, 6.9, 5.7, 25, 15, 14, 10, 2)
go
insert GERMAN_TANKS values
("Pz. Kpfw. I kl.", " Light Command Tank", 6, 14.6, 6.9, 6.6, 32, 15, 15, 15, 3)
go
insert GERMAN_TANKS values
("Pz. Kpfw. II", "Light Tank",  9, 15.2, 7.4, 6.5, 30, 15, 20, 15, 3)
go
insert GERMAN_TANKS values
("Pz. Kpfw. III", "Medium Tank", 21, 17.9, 9.7, 8.3, 28, 15, 30, 30, 5)
go
insert GERMAN_TANKS values
("Pz. Kpfw. III Aus J", "Medium Tank", 22, 17.9, 9.8, 8.3, 28, 15, 50, 30, 5)
go
insert GERMAN_TANKS values
("Pz. Kpfw. IV", "Medium Tank", 24, 19.4, 9.7, 8.6, 28, 15, 50, 30, 5)
go
insert GERMAN_TANKS values
("Pz. Kpfw. V (Panther)", "Heavy Tank", 50, 22.7,11.3, 9.6, 30, 15, 100, 45, 5)
```

```
go
insert GERMAN_TANKS values
("Pz. Kpfw. VI (Tiger)", "Heavy Tank", 60, 20.8, 12.3, 9.4, 25, 15, 102, 80, 5)
go
insert GERMAN_TANKS values
("Tiger (Redesigned)", "Heavy Tank", 75, 24, 12.1, 10.2, 23.6, 10, 150, 80, 5)
go
insert GERMAN_TANKS values
("King Tiger", "Heavy Tank", 75, 23.9, 12.7, 10.2, 23.6, 10, 150, 80, 5)
go
insert GERMAN_TANKS values
("Pz. Jag. Panther", "S.P. Assault Gun", 45, 28.4, 10.9, 9.9, 34, 15, 80, 45, 5 )
go
insert GERMAN_TANKS values
("Pz. Jag. Tiger", "Self-Propelled Gun", 77, 23.9, 12.7, 9.3, 23.6, 10, 250, 80,
6)
go
insert GERMAN_TANKS values
("Sturmgeschutz", "S.P. Assault Gun", 21, 17.9, 9.7, 6.5, 28, 15, 50, 30, 4)
go
insert GERMAN_TANKS values
("Gw. II (Wasp)", "S.P. Light Howitzer", 12, 15.9, 7.4, 7.9, 24, 0, 10, 10, 4)
go
insert GERMAN_TANKS values
("Pz. Kpfw. III Fl.", "Flamethrower Tank", 23.8, 17.9, 9.7, 8.3, 28, 15, 30, 30,
5)
go
insert GERMAN_TANKS values
("Pz. Jag. I", "S.P. Antitank Gun", 8.4, 14.6, 6.9, 7, 26, 15, 15, 10, 4)
go
```

The following data was entered for the table Marines.

```
CREATE TABLE MARINES (
 SHIP        char(20),
 COMMANDER   char(10),
 COMPLEMENT  float,
 NOTES       char(40))
go

insert MARINES VALUES
("Hood ",  "Jones", 100, "Landing Support Forces")
go
insert MARINES VALUES
("Arizona", "Smith", 150, "Special Demolition Forces")
go
insert MARINES VALUES
("Omaha",  "Meyers", 150, "Rangers")
go
insert MARINES VALUES
("Essex",  "Johnson", 60, "Special Demolition Forces")
go
```

```
insert MARINES VALUES
("Shrike", "Whiting", 100, "Commando Forces")
go
```

The following data was entered for the table Planes.

```
CREATE TABLE PLANES (
 MODEL            char(6),
 NAME             char(15),
 TYPE             char(7),
 SUBTYPE          char(7),
 NATION           char(5),
 MANUFACTURER     char(17),
 WINGSPAN         float,
 LENGTH           float,
 HEIGHT           float)
go
insert PLANES VALUES
("P-38","Lightning","Fighter","","USA","Lockheed",52.92,37.83,12.83)
go
insert PLANES VALUES
("P-39","Airacobra","Fighter","Pursuit","USA","Bell",34.00,30.17,11.83)
go
insert PLANES VALUES
("P-40","Warhawk","Fighter","","USA","Curtis-Wright",37.33,31.17,11.83)
go
insert PLANES VALUES
("P-47","Thunderbolt","Fighter","","USA","Republic",40.92,36.25,14.58)
go
insert PLANES VALUES
("P-51","Mustang","Fighter","","USA","North American",37.00,32.25,13.67)
go
insert PLANES VALUES
("","Spitfire","Fighter","","GB","Supermarine",36.83,32.67,12.67)
go
insert PLANES VALUES
("","Hurricane","Fighter","","GB","Hawker",40.08,32.17,13.08)
go
insert PLANES VALUES
("","Typhoon","Fighter","","GB","Hawker",41.67,32.00,14.83)
go
insert PLANES VALUES
("","Tempest","Fighter","","GB","Hawker",41.08,33.75,16.08)
go
insert PLANES VALUES
("Me-109","Me-109","Fighter","","GER","Messerschmitt",32.50,29.00,08.17)
go
insert PLANES VALUES
("Me-110","Me-110","Fighter","","GER","Messerschmitt",53.33,39.58,13.65)
go
insert PLANES VALUES
```

```
("A-20","Havoc","Bomber","Light","USA","Douglas",61.33,48.00,17.83)
go
insert PLANES VALUES
("A-26","Invader","Bomber","Light","USA","Douglas",70.08,50.08,18.50)
go
insert PLANES VALUES
("A-30","Baltimore","Bomber","Light","USA","Martin",61.42,48.58,17.92)
go
insert PLANES VALUES
("A-35","Vengence","Bomber","Dive","USA","Vultee",37.33,31.17,10.83)
go
insert PLANES VALUES
("B-25","Mitchell","Bomber","Medium","USA","North
American",67.92,51.08,15.75)
go
insert PLANES VALUES
("B-26","Marauder","Bomber","Medium","USA","Martin",71.17,58.33,21.50)
go
insert PLANES VALUES
("B-17","Flying
Fortress","Bomber","Heavy","USA","Boeing",103.75,74.75,19.08)
go
insert PLANES VALUES
("B-
24","Liberator","Bomber","Heavy","USA","Consolidated",110.00,67.33,18.00)
go
insert PLANES VALUES
("B-
29","Superfortress","Bomber","Heavy","USA","Boeing",141.58,99.25,29.67)
go
insert PLANES VALUES
("","Wellington","Bomber","Medium","GB","Vickers-
Armstrong",86.33,60.92,17.42)
go
insert PLANES VALUES
("","Halifax","Bomber","Heavy","GB","Handley-Pope",98.83,71.08,20.75)
go
insert PLANES VALUES
("","Stirling","Bomber","Heavy","GB","Short",99.08,87.25,22.75)
go
insert PLANES VALUES
("","Lancaster","Bomber","Heavy","GB","Avro",102.25,69.67,20.00)
go
insert PLANES VALUES
("","Beaufighter","Bomber","Light","GB","Bristol",57.92,41.75,15.83)
go
```

The following data was entered for the schema planes.

```
CREATE TABLE PLANES (
MODEL          char(6),
NAME           char(15),
```

```
TYPE            char(7),
SUBTYPE         char(7),
NATION          char(5),
MANUFACTURER    char(17),
WINGSPAN        float,
LENGTH          float,
HEIGHT          float)
go
insert PLANES VALUES
("P-38","Lightning","Fighter","","USA","Lockheed",52.92,37.83,12.83)
go
insert PLANES VALUES
("P-39","Airacobra","Fighter","Pursuit","USA","Bell",34.00,30.17,11.83)
go
insert PLANES VALUES
("P-40","Warhawk","Fighter","","USA","Curtis-Wright",37.33,31.17,11.83)
go
insert PLANES VALUES
("P-47","Thunderbolt","Fighter","","USA","Republic",40.92,36.25,14.58)
go
insert PLANES VALUES
("P-51","Mustang","Fighter","","USA","North American",37.00,32.25,13.67)
go
insert PLANES VALUES
("","Spitfire","Fighter","","GB","Supermarine",36.83,32.67,12.67)
go
insert PLANES VALUES
("","Hurricane","Fighter","","GB","Hawker",40.08,32.17,13.08)
go
insert PLANES VALUES
("","Typhoon","Fighter","","GB","Hawker",41.67,32.00,14.83)
go
insert PLANES VALUES
("","Tempest","Fighter","","GB","Hawker",41.08,33.75,16.08)
go
insert PLANES VALUES
("Me-109","Me-109","Fighter","","GER","Messerschmitt",32.50,29.00,08.17)
go
insert PLANES VALUES
("Me-110","Me-110","Fighter","","GER","Messerschmitt",53.33,39.58,13.65)
go
insert PLANES VALUES
("A-20","Havoc","Bomber","Light","USA","Douglas",61.33,48.00,17.83)
go
insert PLANES VALUES
("A-26","Invader","Bomber","Light","USA","Douglas",70.08,50.08,18.50)
go
insert PLANES VALUES
("A-30","Baltimore","Bomber","Light","USA","Martin",61.42,48.58,17.92)
go
insert PLANES VALUES
("A-35","Vengence","Bomber","Dive","USA","Vultee",37.33,31.17,10.83)
go
insert PLANES VALUES
```

```
("B-25","Mitchell","Bomber","Medium","USA","North
American",67.92,51.08,15.75)
go
insert PLANES VALUES
("B-26","Marauder","Bomber","Medium","USA","Martin",71.17,58.33,21.50)
go
insert PLANES VALUES
("B-17","Flying
Fortress","Bomber","Heavy","USA","Boeing",103.75,74.75,19.08)
go
insert PLANES VALUES
("B-
24","Liberator","Bomber","Heavy","USA","Consolidated",110.00,67.33,18.00)
go
insert PLANES VALUES
("B-
29","Superfortress","Bomber","Heavy","USA","Boeing",141.58,99.25,29.67)
go
insert PLANES VALUES
("","Wellington","Bomber","Medium","GB","Vickers-
Armstrong",86.33,60.92,17.42)
go
insert PLANES VALUES
("","Halifax","Bomber","Heavy","GB","Handley-Pope",98.83,71.08,20.75)
go
insert PLANES VALUES
("","Stirling","Bomber","Heavy","GB","Short",99.08,87.25,22.75)
go
insert PLANES VALUES
("","Lancaster","Bomber","Heavy","GB","Avro",102.25,69.67,20.00)
go
insert PLANES VALUES
("","Beaufighter","Bomber","Light","GB","Bristol",57.92,41.75,15.83)
go
```

The following data was entered for the table Ships.

```
CREATE TABLE SHIPS (
 NAME         char(20),
 TYPE         char(20),
 NATION       char(10),
 ENGINES      char(5),
 LENGTH       float,
 BEAM         float,
 DISPLACEMENT  float)
go
insert SHIPS VALUES
("Yamamoto","Battleship", "Japan",  "NPX", 863, 127, 64000)
go
insert SHIPS VALUES
("Bismark", "Battleship", "Germany","JR1", 900, 140, 27000)
```

```
go
insert SHIPS VALUES
("Hood ",   "Battleship", "USA",    "SK1", 820, 119, 45000)
go
insert SHIPS VALUES
("Arizona", "Battleship", "USA",    "ND1", 831, 122, 45000)
go
insert SHIPS VALUES
("Omaha",   "Battleship", "USA",    "ND1", 835, 120, 47000)
go
insert SHIPS VALUES
("Essex",   "Carrier",    "USA",    "ND3", 820, 147, 27000)
go
insert SHIPS VALUES
("Shrike",  "Crusier",    "Britain","NS2", 650, 80, 19000)
go
insert SHIPS VALUES
("Vincent", "Destroyer",  "Britain","NS1", 500, 60, 12000)
go
```

## CD ROM Test Data

SPS was unable to locate a really good CD ROM in UNIX format to use in the ATOMS test data. The CD ROM supplied with the drive contains various binary, text, and raster image files which can be modeled in ATOMS as UNIX_FILESYSTEM databases. The CD ROM is treated by the SUN UNIX as a large read only disk drive. Consequently, its contents appear as a standard tree of UNIX directories. No special code was needed to integrate the CD ROM. The real issue when it comes to CD ROM is to build data servers which understand the special formats in which most CD ROM data (such as maps) is encoded and to display or otherwise make it available to ATOMS. This can be a significant undertaking in itself for complicated compressed and encoded formats. The availability of existing code which can be adapted for use in ATOMS would significantly reduce the cost.

The CD ROM local schema models one directory which is on the test CD ROM. This allows users to retrieve the names, sizes in bytes, and dates and times of creation of the files which match the file specification in the entity type in the local schema. Because ATOMS can not interpret any file types other than XBM (x bitmap) and VOX (dialogic voice files) the CD ROM local schema essentially provides the same information which performing a directory command (ls -l) from UNIX would. ATOMS does demonstrate that it is possible to access information from a CD ROM transparently and remotely, as the CD ROM could be located on any machine which was running the ATOMS_FILE_SERVER data server.

## Dialogic Test Data

Two directories containing digitized voice files are provided as part of the test data. These directories are modeled by a local schema as UNIX_FILESYSTEMS. When the view entities which map down to them are retrieved, then entity instances with file(dialogic) attributes are retrieved. Selecting Play Message sends a message to the DIALOGIC_SERVER asking it to play the file whose name is stored in the file(dialogic) attribute. Note that it is important to provide the full path name of the file to be played unless it is in the same directory as the DIALOGIC_SERVER, and therefore the local schema uses the FULL_NAME attribute as the source of the file(dialogic) attribute.

The program DLRECORD in the /aae/atoms/dialogic directory can be used to record additional voice messages. These messages are stored in the directory /VOX by the program, and can be moved where ever desired once they are recorded. The program provides instructions on making new recordings.

The DIALOGIC_SERVER goes directly through an acoustical coupler into the phone provided with the AAE testbed. Software was written by subcontractor CSI to dial an arbitrary number through a standard PBX and play a message file once a confirmation number was entered. This capability was not utilized by the current DIALOGIC_SERVER due to development time constraints, but it could be incorporated at a later date.

## Image Test Data

A directory containing scanned images in XBM (X bitmap) format forms part of the test data. It is modeled as a UNIX_FILESYSTEM by a local schema. When the view entity which maps down this local schema is retrieved, then entity instances with file(XBM) attributes are retrieved. Selecting Display Graphic displays the file whose name is stored in the file(XBM) attribute. Note that it is important to provide the full path name of the file to be displayed unless it is in the same directory as RUN_ATOMS, and therefore the local schema uses the FULL_NAME attribute as the source of the file(XBM) attribute. Attempting to display a file without the file path name could result in a X Windows error which will shutdown the ATOMS session.

The image data could also have been modeled by a Sybase or Oracle database. The full path name of an image file could be stored in a field. The row of data about a particular item could include a file(XBM) attribute which holds the name of the image of the item. This would allow a direct coupling between the relational and image data pertaining to an item.

## *MISSION*

## *OF*

## *ROME LABORATORY*

*Rome Laboratory plans and executes an interdisciplinary program in research, development, test, and technology transition in support of Air Force Command, Control, Communications and Intelligence ($C^3I$) activities for all Air Force platforms. It also executes selected acquisition programs in several areas of expertise. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of $C^3I$ systems. In addition, Rome Laboratory's technology supports other AFSC Product Divisions, the Air Force user community, and other DOD and non-DOD agencies. Rome Laboratory maintains technical competence and research programs in areas including, but not limited to, communications, command and control, battle management, intelligence information processing, computational sciences and software producibility, wide area surveillance/sensors, signal processing, solid state sciences, photonics, electromagnetic technology, superconductivity, and electronic reliability/maintainability and testability.*